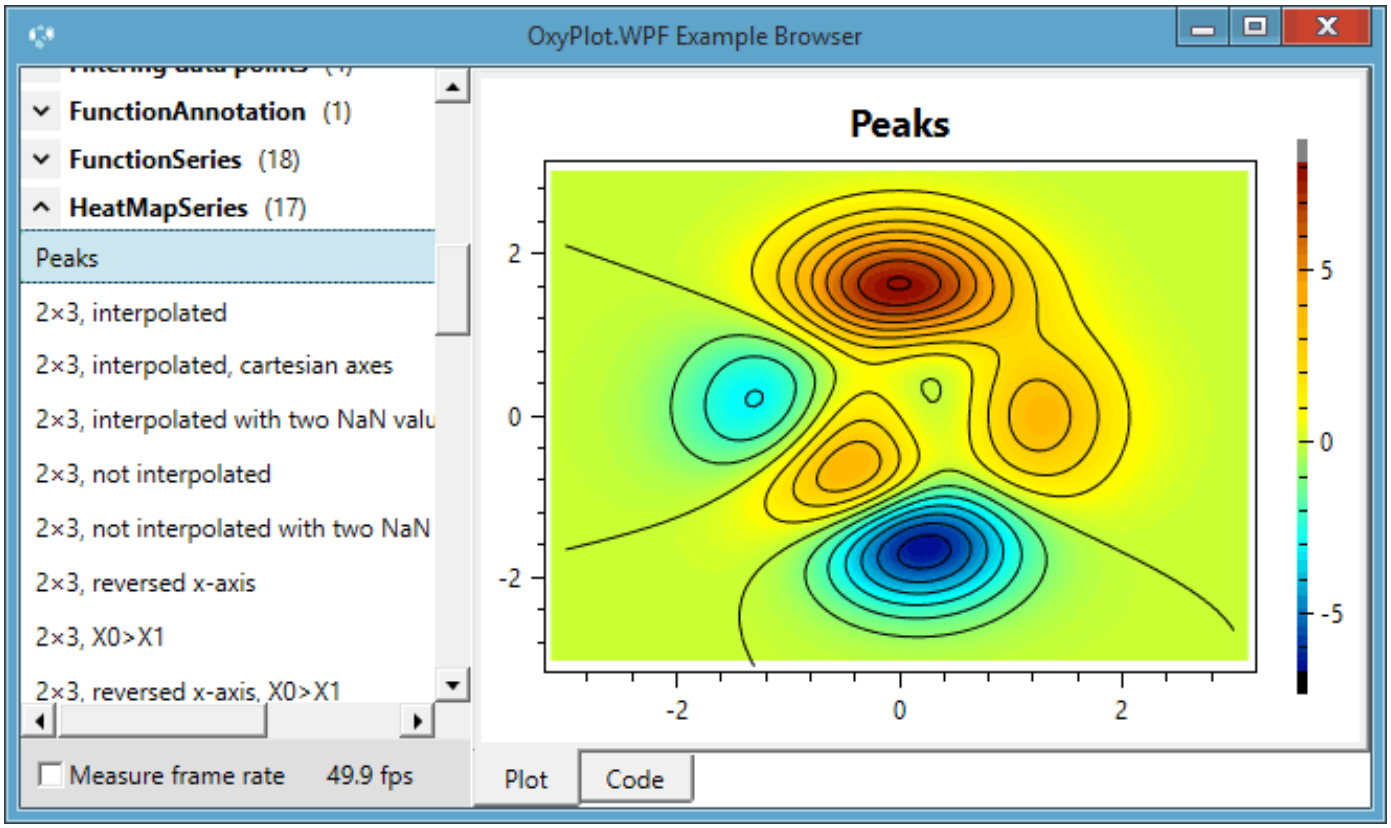
عنوان: **معرفی کتابخانه‌ی OxyPlot**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۷/۲۳ ۱۶:۰
آدرس: www.dotnettips.info
گروه‌ها: WPF, chart, OxyPlot

برای ترسیم نمودار در برنامه‌های WPF، چندین کتابخانه‌ی سورس باز مانند Dynamic Data Display ، Sparrow Toolkit ، GraphIT و ... OxyPlot وجود دارند. در بین این‌ها، کتابخانه‌ی OxyPlot دارای این مزایا است:

- دارای مجوز MIT است. (مجاز هستید از آن در هر نوع برنامه‌ای استفاده کنید)
- cross-platform است. به این معنا که دات نت، WinRT و Xamarin را به خوبی پشتیبانی می‌کند.
- WPF و همچنین WinForms تا Xamarin.Android را پوشش می‌دهد.
- بسته‌های اصلی NuGet آن تا به امروز نزدیک به 40 هزار بار دریافت شده‌اند.
- انجمن فعالی دارد .
- بسیار بسیار غنی است. تا حدی که مرور سطحی مجموعه مثال‌های آن شاید چند ساعت وقت را به خود اختصاص دهد.
- طراحی آن به نحوی است که با الگوی MVVM کاملا سازگاری دارد.
- به صورت متناوبی به روز شده و نگهداری می‌شود.



این برنامه (تصویر فوق) که حاوی مرورگر مثال‌های آن است، در پوشه‌ی Source\Examples\WPF\ExampleBrowser سورس‌های آن قرار دارد.

در ادامه نگاهی خواهیم داشت به نحوه‌ی استفاده از OxyPlot در برنامه‌های WPF جهت رسم نموداری بلادرنگ که اطلاعات آن در زمان اجرای برنامه تهیه شده و در همین حین نیز تغییر می‌کنند.

دریافت بسته‌های نیوگت OxyPlot

برای دریافت دو بسته‌ی OxyPlot.Core و OxyPlot.Wpf تنها کافی است دستور ذیل را در کنسول پاورشل نیوگت اجرا کنیم:

```
PM> install-package OxyPlot.Wpf
```

افزودن تعاریف چارت به View

```
<Window x:Class="OxyPlotWpfTests.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:oxy="http://oxyplot.org/wpf"
        xmlns:oxyPlotWpfTests="clr-namespace:OxyPlotWpfTests"
        Title="MainWindow" Height="350" Width="525">
    <Window.Resources>
        <oxyPlotWpfTests:MainWindowViewModel x:Key="MainWindowViewModel" />
    </Window.Resources>
    <Grid DataContext="{Binding Source={StaticResource MainWindowViewModel}}">
        <oxy:PlotView Model="{Binding PlotModel}"/>
    </Grid>
</Window>
```

ابتدا باید فضای نام oxy اضافه شود. پس از آن oxy:PlotView به صفحه اضافه شده و سپس Model آن از ViewModel برنامه تغذیه می‌گردد.

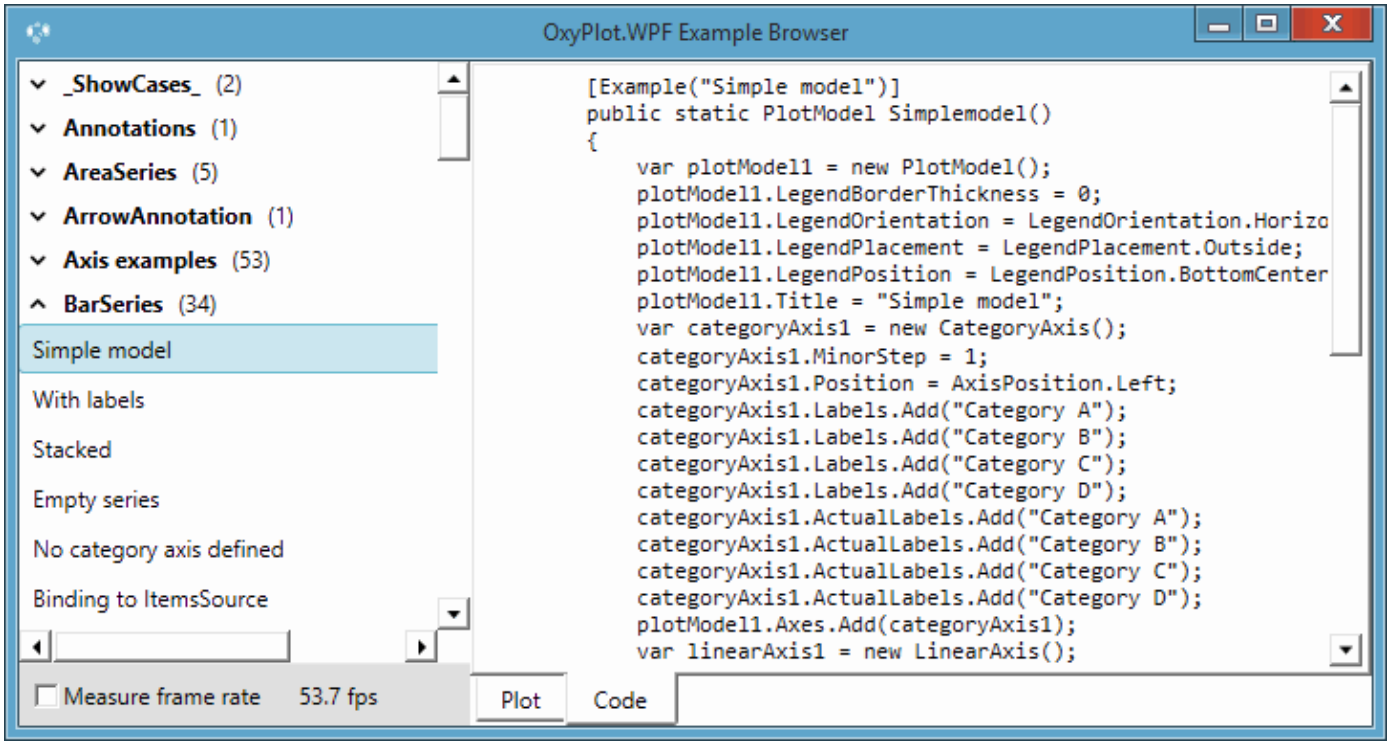
ساختار کلی ViewModel برنامه

کار ViewModel متصل شده به View فوق، مقدار دهی PlotModel است.

```
public class MainWindowViewModel
{
   public PlotModel PlotModel { get; set; }
```

یک نکته‌ی کاربردی

اگر هیچ ایده‌ای نداشتید که این PlotModel را چگونه باید مقدار دهی کرد، به همان برنامه‌ی ExampleBrowser ابتدای مطلب مراجعه کنید.



مثال‌های اجرای شده‌ی آن یک برگه‌ی نمایشی و یک برگه‌ی Code دارند. خروجی این متدها را اگر به خاصیت PlotModel فوق انتساب دهید ... یک چارت کامل خواهید داشت.

مراحل ساخت یک PlotModel

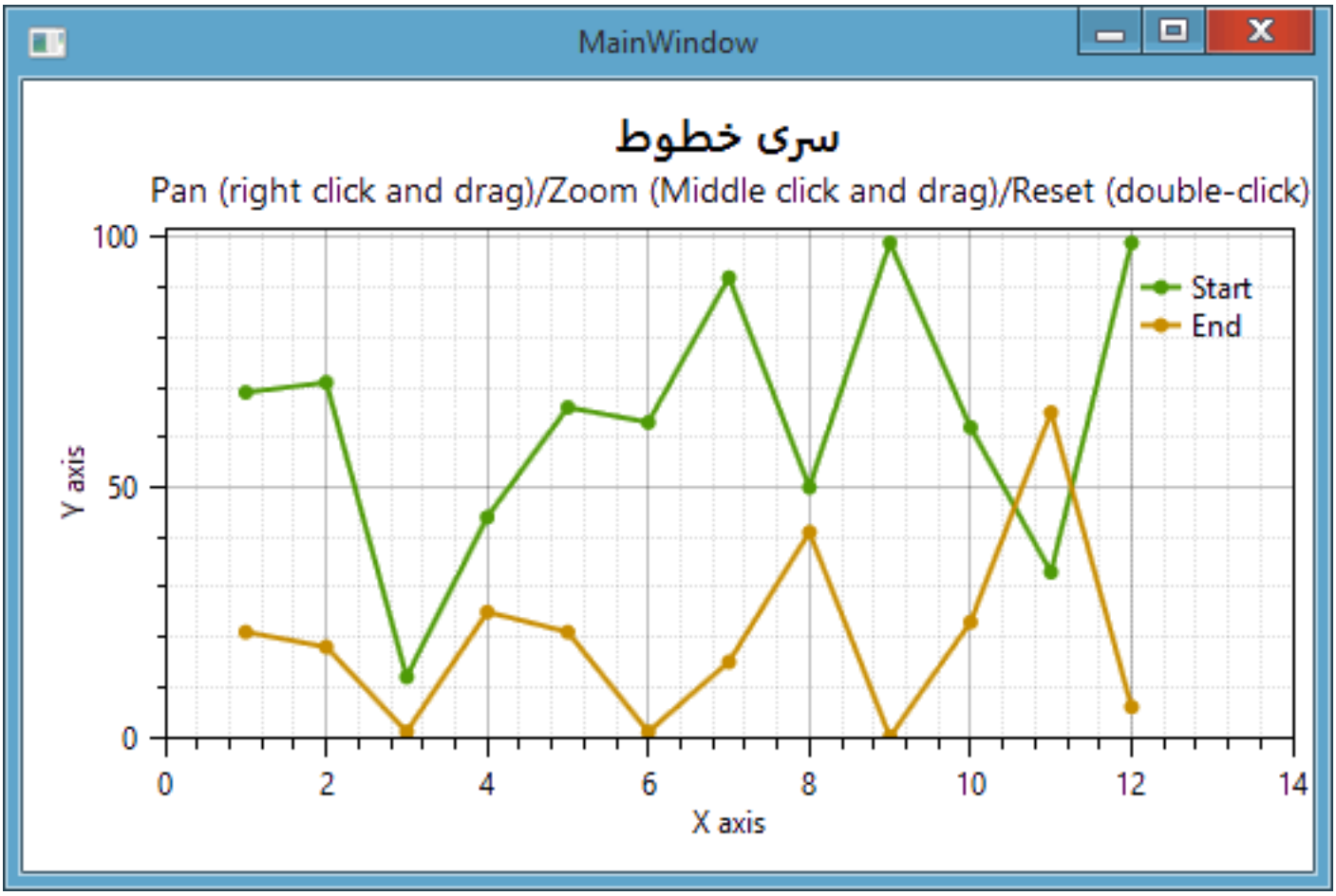
ابتدا نیاز است یک وهله‌ی جدید از PlotModel را ایجاد کنیم:

```
private void createPlotModel()
        {
            PlotModel = new PlotModel
            {
                Title = "سری خطوط",
                Subtitle = "Pan (right click and drag)/Zoom (Middle click and drag)/Reset (double-
click)"
            };
            PlotModel.MouseDown += (sender, args) =>
            {
                if (args.ChangedButton == OxyMouseButton.Left && args.ClickCount == 2)
                {
                    foreach (var axis in PlotModel.Axes)
                        axis.Reset();

                    PlotModel.InvalidatePlot(false);
                }
            };
        }
```

PlotModel در برگیرنده‌ی محورها، نقاط و تمام ناحیه‌ی چارت است. در اینجا عنوان و زیرعنوان نمودار، مقدار دهی شده‌اند.
همچنین در همین ViewModel بدون نیاز به مراجعه به View، می‌توان به رخدادهای مختلف OxyPlot دسترسی داشت. برای مثال می‌خواهیم اگر کاربر دو بار بر روی چارت کلیک کرد، کلیه اعمال zoom و pan آن به حالت اول برگردانده شوند.
برای pan، کافی است دکمه‌ی سمت راست ماوس را نگه داشته و بکشید. به این ترتیب می‌توانید نمودار را بر روی محورهای X و Y حرکت دهید.
برای zoom نیاز است دکمه‌ی وسط ماوس را نگه داشته و بکشید. ناحیه‌ای که در این حالت نمایان می‌گردد، محل بزرگنمایی نهایی

خواهد بود.
این دو قابلیت به صورت توکار در OxyPlot قرار دارند و نیازی به کدنویسی برای فعال سازی آن‌ها نیست.



افزودن محورهای X و Y

محور X در مثال ما، از نوع LinearAxis است. بهتر است متغیر آن‌را در سطح کلاس تعریف کرد تا بتوان از آن در سایر قسمت‌های چارت نیز بهره گرفت:

```
readonly LinearAxis _xAxis = new LinearAxis();
        private void addXAxis()
        {
            _xAxis.Minimum = 0;
            _xAxis.MaximumPadding = 1;
            _xAxis.MinimumPadding = 1;
            _xAxis.Position = AxisPosition.Bottom;
            _xAxis.Title = "X axis";
            _xAxis.MajorGridlineStyle = LineStyle.Solid;
            _xAxis.MinorGridlineStyle = LineStyle.Dot;
            PlotModel.Axes.Add(_xAxis);
        }
```

در اینجا مقدار خاصیت Position، مشخص می‌کند که این محور در کجا باید قرار گیرد. اگر مقدار دهی نشود، محور Y را تشکیل خواهد داد.
مقدار دهی GridlineStyleها سبب ایجاد یک Grid خاکستری در نمودار می‌شوند.
در آخر نیاز است این محور به محورهای PlotModel اضافه شود.

تعریف محور Y نیز به همین نحو است. اگر مقدار خاصیت Position ذکر نشود، این محور در سمت چپ صفحه قرار می‌گیرد:

```
readonly LinearAxis _yAxis = new LinearAxis();
        private void addYAxis()
        {
            _yAxis.Minimum = 0;
            _yAxis.Title = "Y axis";
            _yAxis.MaximumPadding = 1;
            _yAxis.MinimumPadding = 1;
            _yAxis.MajorGridlineStyle = LineStyle.Solid;
            _yAxis.MinorGridlineStyle = LineStyle.Dot;
            PlotModel.Axes.Add(_yAxis);
        }
```

افزودن تعاریف سری‌های خطوط

در تصویر فوق، دو سری خط را ملاحظه می‌کنید. تعاریف پایه سری اول آن به این صورت است:

```
readonly LineSeries _lineSeries1 = new LineSeries();
        private void addLineSeries1()
        {
            _lineSeries1.MarkerType = MarkerType.Circle;
            _lineSeries1.StrokeThickness = 2;
            _lineSeries1.MarkerSize = 3;
            _lineSeries1.Title = "Start";
            _lineSeries1.MouseDown += (s, e) =>
            {
                if (e.ChangedButton == OxyMouseButton.Left)
                {
                    PlotModel.Subtitle = "Index of nearest point in LineSeries: " +
Math.Round(e.HitTestResult.Index);
                    PlotModel.InvalidatePlot(false);
                }
            };
            PlotModel.Series.Add(_lineSeries1);
        }
```

مقدار خاصیت MarkerType، نحوه‌ی نمایش نقاط اضافه شده را مشخص می‌کند. خاصیت Title، عنوان آن‌را که در کنار صفحه نمایش داده شده، تعیین کرده و در آخر، این سری نیز باید به سری‌های PlotModel اضافه گردد.
هر سری دارای خاصیت MouseDown نیز هست. برای مثال اگر علاقمندید که کلیک کاربر بر روی نقاط مختلف را دریافت کرده و سپس بر این اساس، اطلاعات خاصی را نمایش دهید، می‌توانید از مقدار e.HitTestResult.Index استفاده کنید. در اینجا ایندکس نزدیک‌ترین نقطه به محل کلیک کاربر یافت می‌شود.

تعاریف اولیه سری دوم نیز به همین ترتیب هستند:

```
readonly LineSeries _lineSeries2 = new LineSeries();
        private void addLineSeries2()
        {
            _lineSeries2.MarkerType = MarkerType.Circle;
            _lineSeries2.Title = "End";
            _lineSeries2.StrokeThickness = 2;
            _lineSeries2.MarkerSize = 3;
            _lineSeries2.MouseDown += (s, e) =>
            {
                if (e.ChangedButton == OxyMouseButton.Left)
                {
                    PlotModel.Subtitle = "Index of nearest point in LineSeries: " +
Math.Round(e.HitTestResult.Index);
                    PlotModel.InvalidatePlot(false);
                }
            };
            PlotModel.Series.Add(_lineSeries2);
        }
```

به روز رسانی دستی OxyPlot

پس از نمایش اولیه OxyPlot، هر تغییری که در اطلاعات آن صورت گیرد، نمایش داده نخواهد شد. برای به روز رسانی آن فقط کافی است متد PlotModel.InvalidatePlot را فراخوانی نمائید. برای نمونه در متدهای فوق، کلیک ماوس، پس از رسم نمودار انجام می‌شود. بنابراین اگر نیاز است زیرعنوان نمودار تغییر کند، باید متد PlotModel.InvalidatePlot نیز فراخوانی گردد.

ایجاد یک تایمر برای افزودن نقاط به صورت پویا

در ادامه می‌خواهیم نقاطی را به صورت پویا به نمودار اضافه کنیم. نمایش یکباره نمودار، نکته‌ی خاصی ندارد. تنها کافی است توسط lineSeries1.Points.Add یک سری DataPoint را اضافه کنید. این نقاط در زمان نمایش View، به یکباره نمایش داده خواهند شد. اما در اینجا ابتدا یک چارت خالی نمایش داده می‌شود و سپس قرار است نقاطی به آن اضافه شوند.

```
private int _xMax;
        private int _yMax;
        private bool _haveNewPoints;
        private void addPoints()
        {
            var timer = new DispatcherTimer {Interval = TimeSpan.FromSeconds(1)};
            var rnd = new Random();
            var x = 1;
            updateXMax(x);
            timer.Tick += (sender, args) =>
            {
                var y1 = rnd.Next(100);
                updateYMax(y1);
                _lineSeries1.Points.Add(new DataPoint(x, y1));

                var y2 = rnd.Next(100);
                updateYMax(y2);
                _lineSeries2.Points.Add(new DataPoint(x, rnd.Next(y2)));

                x++;

                updateXMax(x);
                _haveNewPoints = true;
            };
            timer.Start();
        }

        private void updateXMax(int value)
        {
            if (value > _xMax)
            {
                _xMax = value;
            }
        }

        private void updateYMax(int value)
        {
            if (value > _yMax)
            {
                _yMax = value;
            }
        }
```

چند نکته در اینجا حائز اهمیت هستند:

- افزودن نقاط جدید توسط متدهای lineSeries1.Points.Add انجام می‌شوند.
- مقادیر max محورهای x و y را نیز ذخیره می‌کنیم. اگر نقاط برنامه پویا نباشند، OxyPlot به صورت خودکار نمودار را با مقیاس درستی ترسیم می‌کند. اما اگر نقاط پویا باشند، نیاز است حداکثر محورهای x و y را به صورت دستی در آن تنظیم کنیم. به همین جهت متدهای updateXMax و updateYMax در اینجا فراخوانی شده‌اند.
- به روز رسانی ظاهر چارت، توسط متد زیر انجام می‌شود:

```
private readonly Stopwatch _stopwatch = new Stopwatch();
        private void updatePlot()
```

```
        {
            CompositionTarget.Rendering += (sender, args) =>
            {
                if (_stopwatch.ElapsedMilliseconds > _lastUpdateMilliseconds + 2000 && _haveNewPoints)
                {
                    if (_yMax > 0 && _xMax > 0)
                    {
                        _yAxis.Maximum = _yMax + 3;
                        _xAxis.Maximum = _xMax + 1;
                    }

                    PlotModel.InvalidatePlot(false);

                    _haveNewPoints = false;
                    _lastUpdateMilliseconds = _stopwatch.ElapsedMilliseconds;
                }
            };
        }
```

کل کاری که در اینجا انجام شده، فراخوانی کنترل شده‌ی PlotModel.InvalidatePlot هر دو ثانیه یکبار است.
CompositionTarget.Rendering بر اساس رندر View، عمل کرده و از آن می‌توان برای به روز رسانی نمایشی چارت استفاده کرد. اگر متد PlotModel.InvalidatePlot را دقیقا در زمان افزودن نقاط فراخوانی کنیم به CPU Usage بالایی خواهیم رسید. به همین جهت نیاز است فراخوانی آن کنترل شده و در فواصل زمانی مشخصی باشد. همچنین اگر نقطه‌ای اضافه نشده (بر اساس مقدار haveNewPoints)، به روز رسانی انجام نخواهد شد.
نکته‌ی دیگری که در متد updatePlot فوق درنظر گرفته شده‌است، تغییر مقدار Maximum محورهای x و y بر اساس حداکثرهای نقاط اضافه شده‌است. به این ترتیب نمودار به صورت خودکار جهت نمایش کل اطلاعات، تغییر اندازه خواهد داد.
البته همانطور که عنوان شد، تمام این تهمیدات برای نمایش نمودارهای بلادرنگ است. اگر کار مقدار دهی Points.Add را فقط یکبار در سازنده‌ی ViewModel انجام می‌دهید، نیازی به این نکات نخواهید داشت.

کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید :

OxyPlotWpfTests.zip

نظرات خوانندگان

نویسنده: رامین علیرضایی
تاریخ: ۱۳۹۳/۰۷/۲۳ ۱۹:۱۱

سلام و تشکر فراوان بابت معرفی این کتابخانه.
با توجه به استفاده این کتابخانه از PdfSharp آیا خروجی PDF آن با زبان فارسی مشکلی نخواهد داشت؟
با سپاس

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۷/۲۳ ۱۹:۴۰

- هسته‌ی آن هیچ وابستگی به کتابخانه‌ی خاصی ندارد.
- PdfSharp از زبان‌های راست به چپ پشتیبانی نمی‌کند. اما ... یک مثال کامل export در اینجا دارد. خروجی تصویر، Svg، Xaml و XPS و امثال آن، مشکلی با زبان فارسی ندارند.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۷/۲۸ ۱۴:۲۰

یک نکته‌ی تکمیلی

نمایش tracker آن با حرکت ماوس، بجای کلیک بر روی نقاط (حالت پیش فرض)

```
private IPlotController _controller;
        public IPlotController Controller
        {
            get
            {
                if (_controller == null)
                {
                    // show tracker with mouse move
                    _controller = new PlotController();
                    _controller.BindMouseEnter(PlotCommands.HoverPointsOnlyTrack);
                }
                return _controller;
            }
        }
```

و بعد

```
Controller="{Binding Controller}"
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۱۰/۰۴ ۱۴:۳۱

مثال این نکته را به همراه نمایش اطلاعات اضافی در tracker آن، از اینجا می‌توانید دریافت کنید: OxyPlotWpfTests2.zip