عنوان: **آشنایی با WPF قسمت دوم: Layouts بخش اول**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۲/۰۸ ۲:۱۵
آدرس: www.dotnettips.info
گروه‌ها: WPF

layoutها یکی از مهمترین قسمت‌های یک برنامه‌ی کاربردی هستند. چیدمان کنترل‌ها روی یک ناحیه با دادن مختصات پیکسلی ثابت، ممکن است در یک محیط محدود خود را خوب نشان بدهد ولی به زودی با تغییر محیط برنامه و یا تغییر وضوح تصویر صفحه نمایش، برنامه از کنترل خارج خواهد شد؛ در نتیجه استفاده از Layoutها یا پنل‌ها در WPF امری حیاتی و مهم هستند. layoutها که با نام container هم شناخته می‌شوند وظیفه دارند که بگویند چه کنترل‌هایی در کجا و چگونه باید در صفحه‌ی برنامه قرار بگیرند.
پنل‌های توکار در WPF به دسته‌های زیر تقسیم می‌شوند:

Grid Panel

Stack Panel

Dock Panel

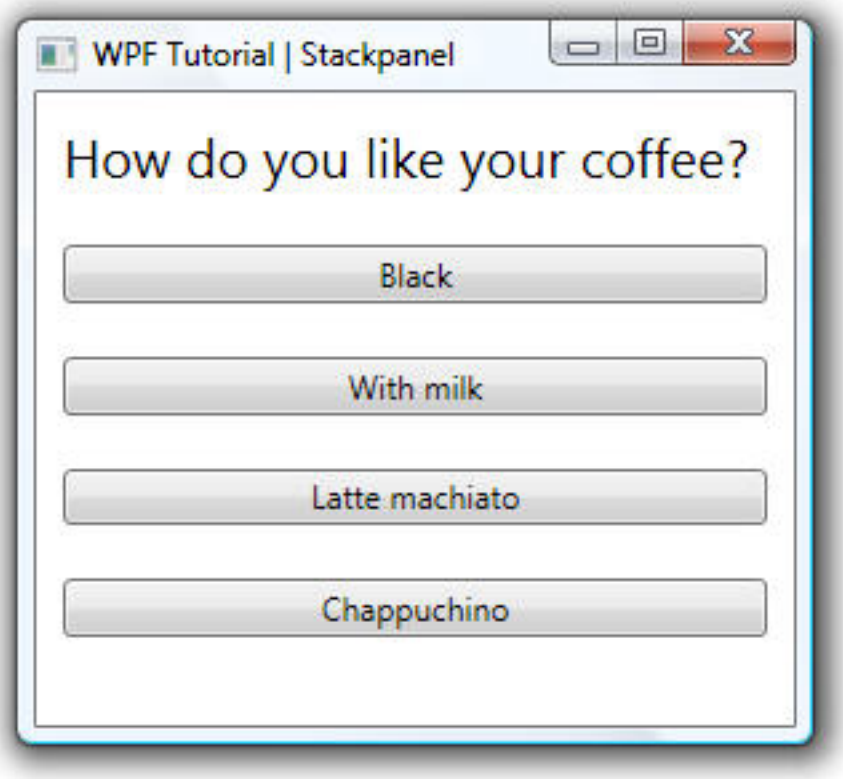
Wrap Panel

Canvas Panel

StackPanel

این پنل یکی از ساده‌ترین و سودمندترین پنل هاست که بر اساس جهت Orientation افقی یا عمودی که به آن تنظیم می‌شود، کنترل‌ها را کنار یکدیگر یا زیر یکدیگر قرار می‌دهد. این کنترل برای ایجاد و تهیه لیست‌های مختلف مناسب است. در ساختار داخلی کنترل‌های لیست و کامبو و منوها که در WPF موجود هستند این پنل استفاده شده است. کد زیر یک نمونه کاربرد Stack Panel را نشان می‌دهد که به صورت عمودی چینش شده است.

```
<StackPanel>
  <TextBlock Margin="10" FontSize="20">How do you like your coffee?</TextBlock>
  <Button Margin="10">Black</Button>
  <Button Margin="10">With milk</Button>
  <Button Margin="10">Latte machiato</Button>
  <Button Margin="10">Chappuchino</Button>
</StackPanel>
```



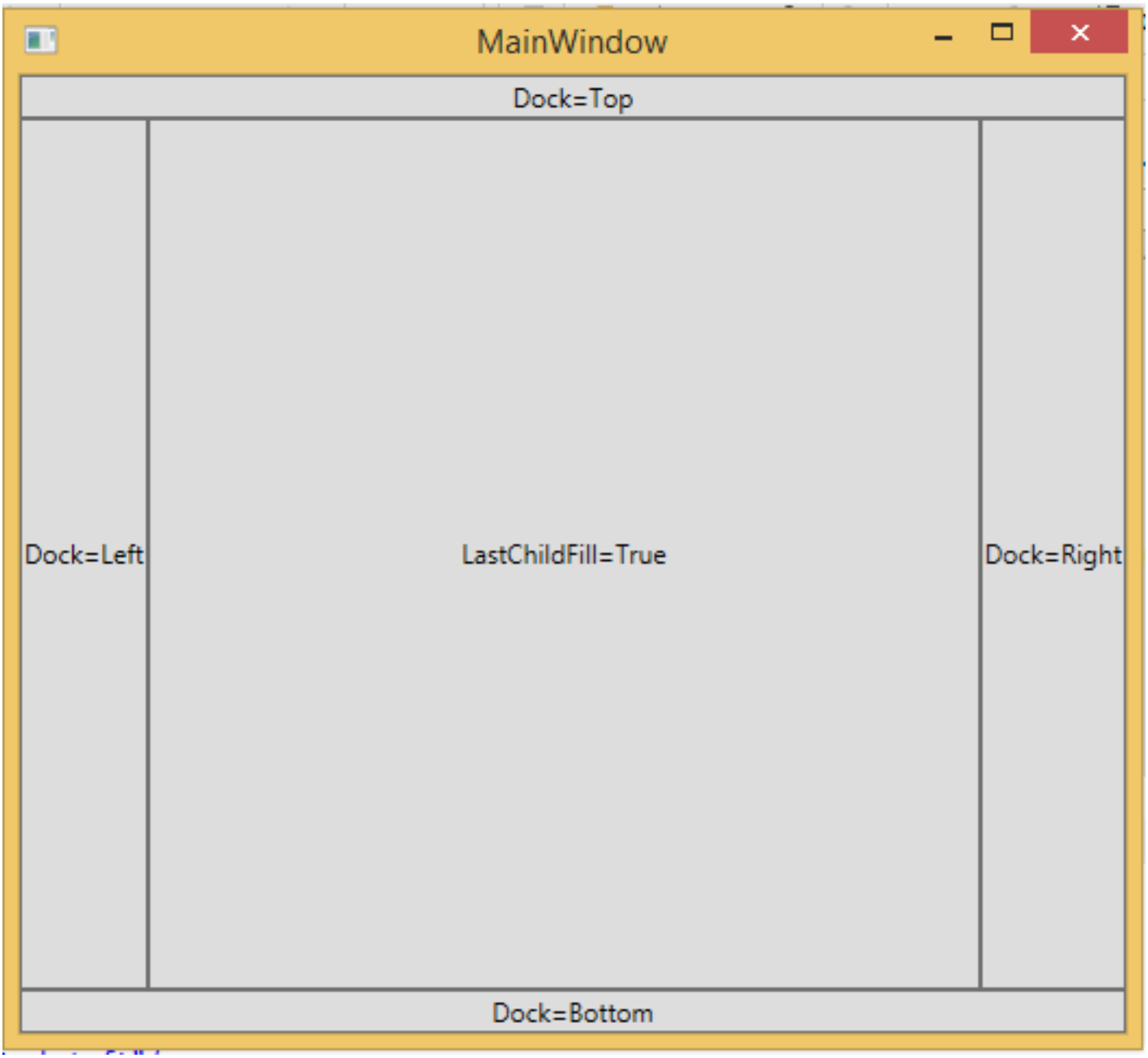
نکته‌ی مهم اینکه میتوانید در اینجا از یک nested layout هم استفاده کنید بدین صورت که یک layout را داخل یک layout دیگر قرار دهید. کد زیر ترکیب دو stack panel را به صورت افقی و عمودی به ما نشان می‌دهد:

```
<StackPanel Orientation="Vertical"> <!-- Vertical is the default -->
  <Label Background="Red">Red 1</Label>
  <Label Background="LightGreen">Green 1</Label>
  <StackPanel Orientation="Horizontal">
    <Label Background="Red">Red 2</Label>
    <Label Background="LightGreen">Green 2</Label>
    <Label Background="LightBlue">Blue 2</Label>
    <Label Background="Yellow">Yellow 2</Label>
    <Label Background="Orange">Orange 2</Label>
  </StackPanel>
  <Label Background="LightBlue">Blue 1</Label>
  <Label Background="Yellow">Yellow 1</Label>
  <Label Background="Orange">Orange 1</Label>
</StackPanel>
```

Dock Panel

احتمالا به خاطر نامش، نحوه کارش را حدس زده اید. این پنل، اشیاء موجود را در 4 جهت و مرکز می‌چسباند. مشخص نمودن جهت چسبیده شدن هر کنترل توسط خاصیت DockPanel.Dock صورت می‌گیرد و مقدار Left، مقدار پیش فرض است. در صورتی که بخواهید المانی را در مرکز بچسبانید باید آن را به عنوان آخرین المان معرفی کرده و در Dock Panel مقدار خاصیت LastChildFill را با True برابر کنید.

```
<DockPanel LastChildFill="True">
    <Button Content="Dock=Top" DockPanel.Dock="Top"/>
    <Button Content="Dock=Bottom" DockPanel.Dock="Bottom"/>
    <Button Content="Dock=Left"/>
    <Button Content="Dock=Right" DockPanel.Dock="Right"/>
    <Button Content="LastChildFill=True"/>
</DockPanel>
```



به نحوه‌ی تعریف خاصیت DockPanel.Dock دقت کنید به این نوع خاصیت‌ها، Attached Dependency Property (شاید در فارسی بتوانیم *خاصیت‌های وابستگی متصل* صدا بزنیم) می‌گویند. این خاصیت‌ها نوع خاصی از خاصیت‌های وابستگی هستند که به شما اجازه می‌دهند مقداری را به شیءایی نسبت دهید که آن شیء چیزی در مورد آن نمی‌داند. بهترین مثال در مورد این ویژگی، پنل‌ها هستند که یکی از موارد استفاده‌ی از آن را در بالا می‌بینید. هر پنل می‌تواند تا بی نهایت المان فرزند داشته باشد که هر المان باید خواصش توسط پنل مشخص گردد. ولی اگر پنل ما تعداد زیادی فرزند داشته باشد، نوشتن خواص هر کدام از فرزندها داخل تگ پنل، کاری غیر ممکن است. اینجاست که این نوع خاصیت‌ها خودشان را نشان می‌دهند. پس به این نحو مقادیر، داخل کنترل هر تگ تعریف می‌شود ولی توسط پنل مورد استفاده قرار می‌گیرد. نحوه‌ی نوشتن این نوع خاصیت: ابتدا یک پیشوند از نوع تگ پنل را در ابتدا آورده و سپس بعد از .(نقطه) نام خاصیت را ذکر می‌کنیم.

نحوه‌ی تعریف این نوع خاصیت‌ها در یک کلاس به صورت زیر است که برای شیء یا پنل canvas می‌باشد:

```
public static readonly DependencyProperty TopProperty =
    DependencyProperty.RegisterAttached("Top",
    typeof(double), typeof(Canvas),
    new FrameworkPropertyMetadata(0d,
```

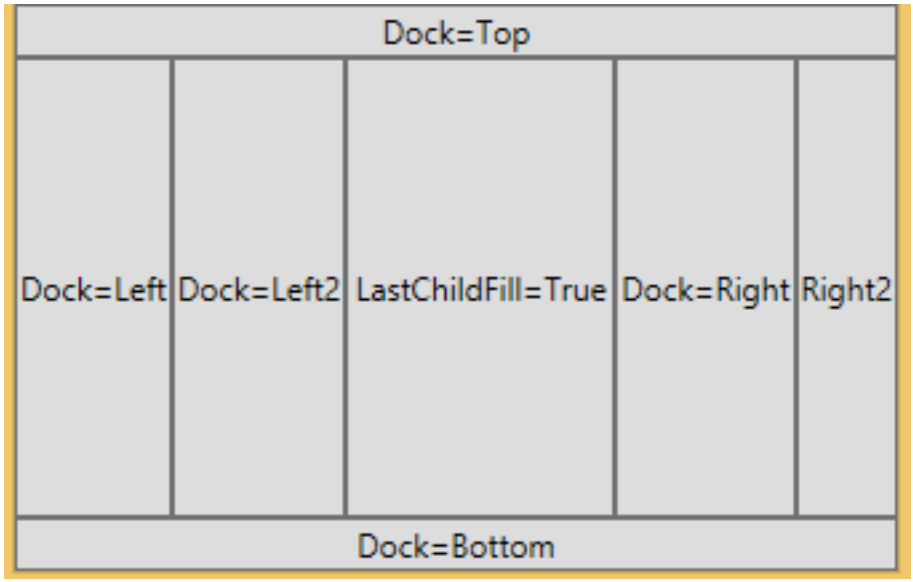
```
        FrameworkPropertyMetadataOptions.Inherits));

public static void SetTop(UIElement element, double value)
{
    element.SetValue(TopProperty, value);
}

public static double GetTop(UIElement element)
{
    return (double)element.GetValue(TopProperty);
}
```

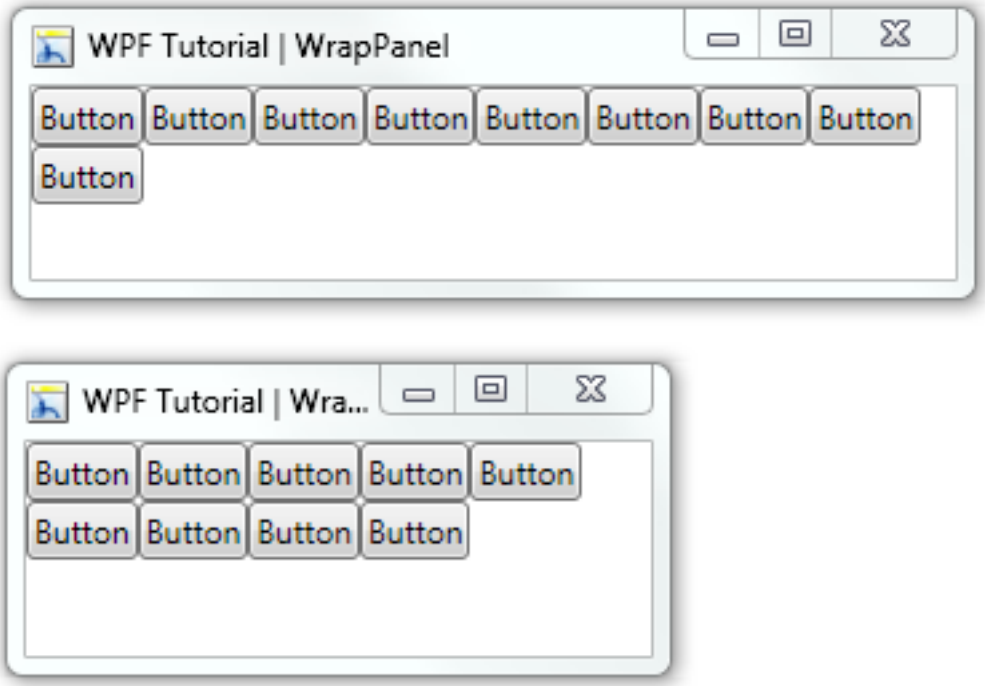
در مثال dockPanel بالا در هر طرف تنها یک المان قرار دادیم. برای قرار دادن المان‌های بیشتر در طرفین، تنها ذکر یک المان جدید با خاصیت Dockpanel.dock کافی است و الویت نمایش آن‌ها بر اساس ترتیب نوشتن تگها توسط شماست. مثال زیر این نکته را نشان می‌دهد:

```
<Button Content="Dock=Top" DockPanel.Dock="Top"/>
        <Button Content="Dock=Bottom" DockPanel.Dock="Bottom"/>
        <Button Content="Dock=Left"/>
        <Button Content="Dock=Left2"/>
        <Button Content="Right2" DockPanel.Dock="Right"/>
        <Button Content="Dock=Right" DockPanel.Dock="Right"/>
        <Button Content="LastChildFill=True"/>
```



Wrap Panel

این پنل بسیار شبیه StackPanel هست ولی مثل آن اشیاء را در یک سطر یا ستون ادامه نمی‌دهد؛ بلکه با رسیدن به انتهای پنجره، سطر یا ستون جدیدی را آغاز می‌کند. در stack panel با پایان پنجره، ادامه اشیا آن قابل مشاهده نبود ولی در این شیء با اتمام و رسیدن به لبه‌ی پنجره، اشیاء در سر جدید (افقی) یا ستون جدید (عمودی) نمایش داده می‌شوند. این پنل‌ها می‌توانند در ساخت تب‌ها و نوار ابزار استفاده شوند.



Canvas Panel

پایه‌ای‌ترین layout موجود در WPF است. موقعیت قرارگیری المان‌های فرزندش بر اساس نقاط تعیین شده است.این پنل بیشتر برای رسم اشکال و گرافیک دو بعدی مناسب است و اصلا برای قرارگیری کنترل‌های WPF روی آن توصیه نمی‌شود و مشکل winformها در آن رخ خواهد داد.

شروع ترسیم یک شکل دو بعدی روی آن بر اساس دوتا از چهار "خاصیت‌های وابستگی متصل" صورت می‌گیرد که به شرح زیر هستند:

Canvas.LEFT

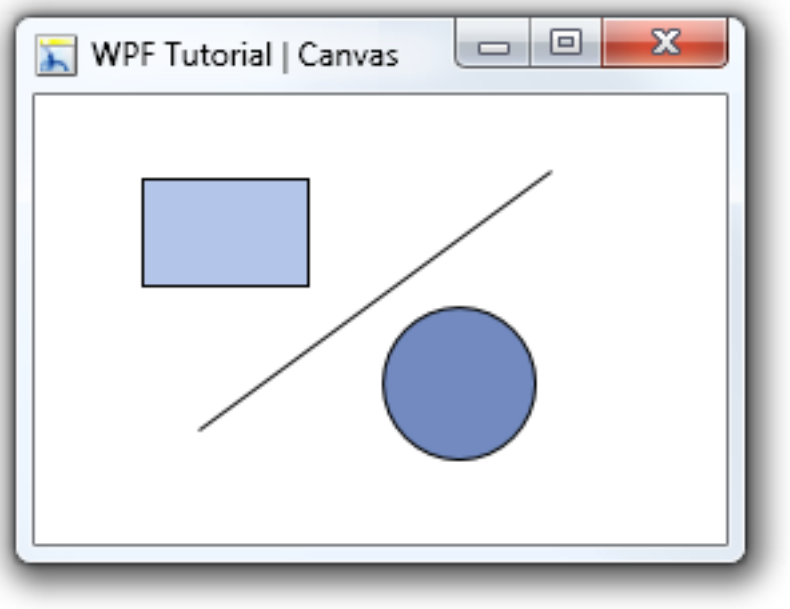
Canvas.RIGHT

Canvas.TOP

Canvas.BOTTOM

نمونه از کد نوشته شده آن به صورت زیر است:

```
<Canvas>
    <Rectangle Canvas.Left="40" Canvas.Top="31" Width="63" Height="41" Fill="Blue"  />
    <Ellipse Canvas.Left="130" Canvas.Top="79" Width="58" Height="58" Fill="Blue"  />
    <Path Canvas.Left="61" Canvas.Top="28" Width="133" Height="98" Fill="Blue"
          Stretch="Fill" Data="M61,125 L193,28"/>
</Canvas>
```



ترتیب قرارگیری اشکال روی هم در canvas به ترتیبی انجام می‌گیرد که در XAML نوشته اید ولی می‌توان با استفاده از خاصیت Canvas.ZIndex این ترتیب را تغییر داد.

```
<Canvas>
    <Ellipse Fill="Green" Width="60" Height="60" Canvas.Left="30" Canvas.Top="20"
             Canvas.ZIndex="1"/>
    <Ellipse Fill="Blue"  Width="60" Height="60" Canvas.Left="60" Canvas.Top="40"/>
</Canvas>
```

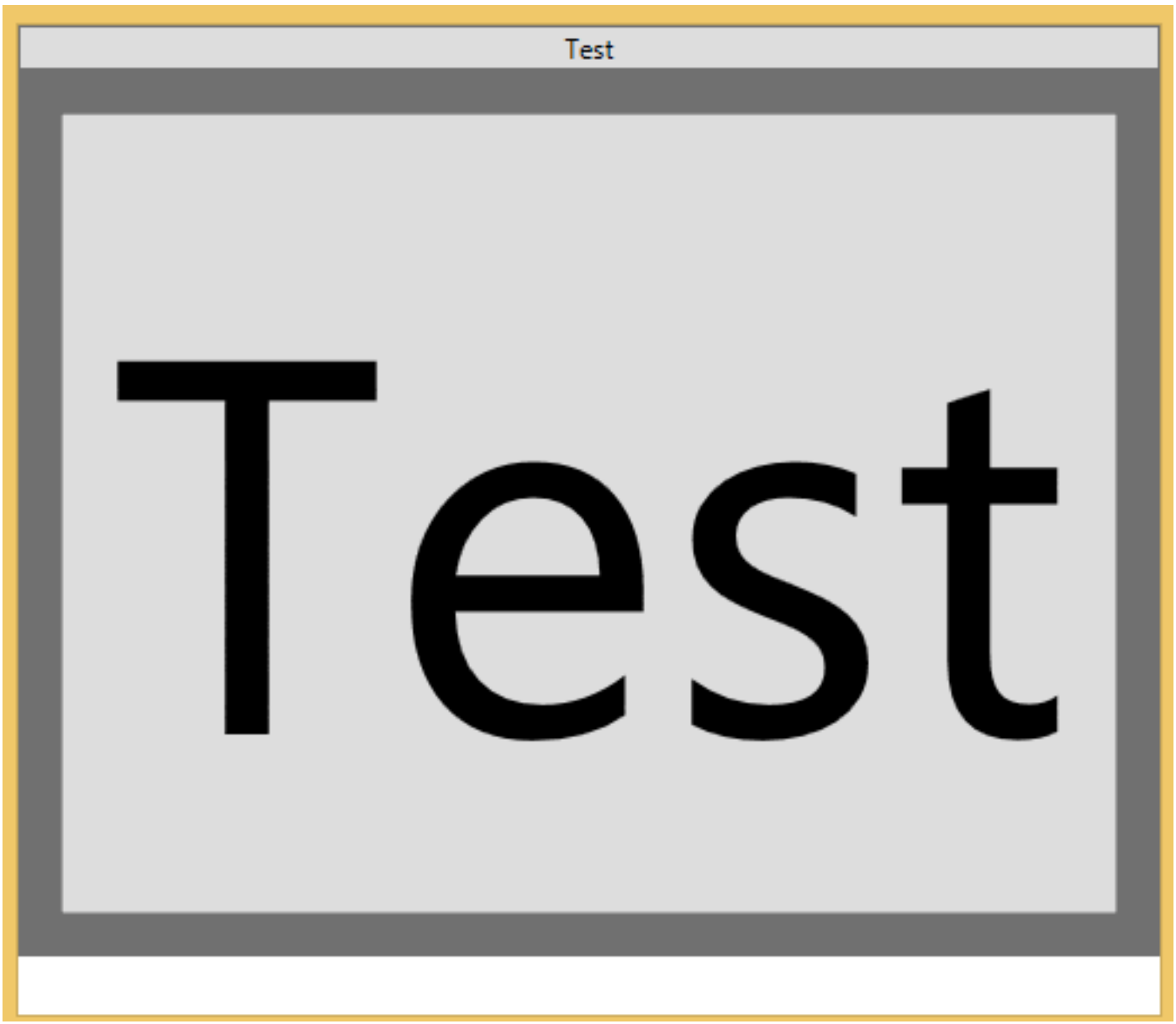
شکل زیر در سمت راست، نتیجه‌ی کد بالاست ولی بدون ذکر ZIndex شکل سمت چپ نتیجه‌ی کار خواهد بود.

ViewBox

شاید عده‌ای به سختی آن را یک Layout بدانند و بیشتر آن را یک کنترل معمولی می‌شناسند ولی وظیفه‌ی آن بسیار شبیه Layout هاست. خصوصیتی که این شیء دارد این است که با تغییر اندازه محیط برنامه به هر نحوی، یک تغییر مقیاس روی اشیاء داخل آن رخ می‌دهد و کنترل‌ها به همراه متون و هر چیزی که در درخت منطقی و بصری است Scale آن تغییر می‌یابند.
نمونه‌ی کد زیر را تست کنید تا تفاوت بین دو Button را ببینید:

```
<StackPanel Orientation="Vertical">
        <Button Content="Test" />
        <Viewbox Stretch="Uniform">
            <Button Content="Test" />
        </Viewbox>
    </StackPanel>
```

نتیجه‌ی کار:



در بخش دوم Layoutها مبحث گرید و ساخت Layout اختصاصی و تعدادی از خاصیت‌ها را بررسی خواهیم کرد.

نظرات خوانندگان

نویسنده: افشین عباسپور
تاریخ: ۱۳۹۴/۰۲/۰۸ ۱۵:۲۶

خیلی ممنون از آموزش مفید و کمیاب WPF ... تا جایی که من میدونم امکان استفاده از کامپوننت‌های WPF در WinApplication وجود دارد . آیا بلعکس این هم امکان پذیر است ؟ استفاده کامپوننت‌های Win Form در Wpf امکان داره ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۲/۰۸ ۱۵:۳۵

بله. WPF برای اینکار دارای WindowsFormsHost هست. (خارج از موضوع بحث جاری است)

نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۲/۰۸ ۱۵:۳۵

ممنون از شما
این صفحه رو مطالعه کنید.