

لغة التصميم

الاصدار الاول



المهندس فيصل الاسود

باحث اكاديمي في جامعة SRM الدولية للعلوم والتكنولوجيا

اول كتاب عربي يقدم محتوى التصميم باستخدام WPF



تقنية WPF

لغة التصميم XAML

المهندس : فيصل الأسود

باحث أكاديمي في جامعة SRM الدولية للعلوم والتكنولوجيا باختصاص هندسة البرمجيات.

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



حول المؤلف:

السلام عليكم اخوتي

انا فيصل الأسود مهندس برمجيات وباحث اكاديمي في جامعة SRM الدولية للعلوم والتكنولوجيا تكرم عليّ الله بأن ادرس هذا المجال الذي لطالما احببته ، اتمنى ان اقدم كل يوم ولو شيء بسيطاً لعالمنا العربي في هذا المجال.

اهدي هذا العمل المتواضع الى ابي وامي وزوجتي التي صبرت على انشغالي كما اقدمه الى اخوتي رفاق دربي والى كل شخص يقدر قيمة العلم ، أتمنى منكم الدعاءواتمنى لكم التوفيق.

كما يمكنكم زيارة قناتي على اليوتيوب والتواصل معي للاستفسار عن أي مسألة .

فيصل الاسود

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



حول الكتاب:

بعد تفوق شبابنا العربي في عدة مجالات أهمها تطبيقات الويب و سطح المكتب لابد لنا ان نضيف لمسة من الفن الى تطبيقاتنا لنجعلها أكثر رقي وسلاسة لذا اقدم لكم هذا الكتاب الذي يحمل في طياته تعليم تقنية الـ WPF والتي تكتب بلغة Xaml لبناء واجهات وصفحات ويب متينة وانيقة ، أتمنى ان ينال اعجابكم.

كما يمكنكم زيارة قناتي على اليوتيوب للحصول على دروس مسجلة لجميع اقسام الكتاب.

المهندس فيصل الأسود
فيصل الأسود

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube





مقدمة إلى WPF

Windows Presentation Foundation

المهندس فيصل الأسود

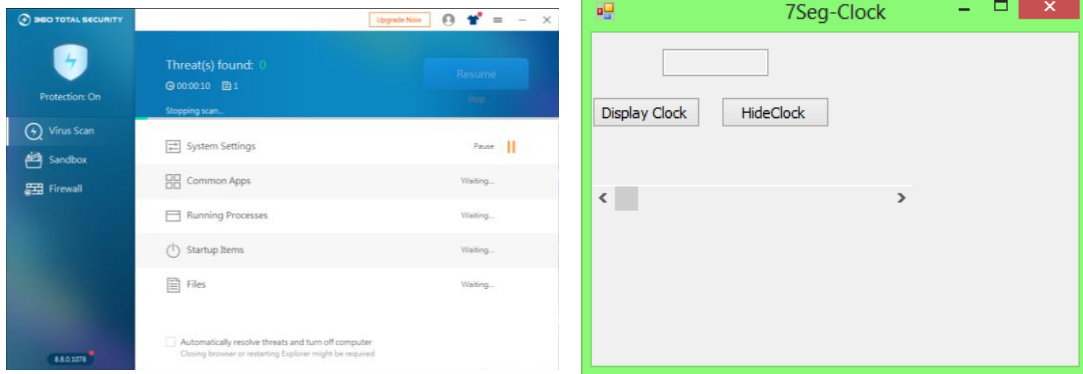
تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



مقدمة الى WPF:

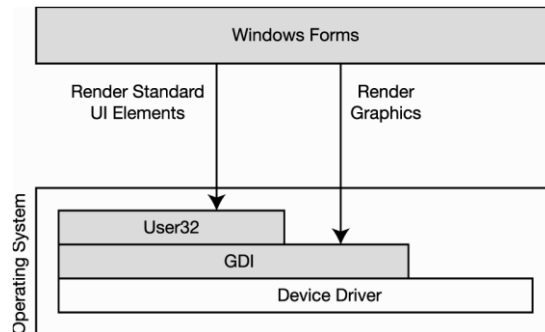
هل نحتاج الى واجهات مستخدم او أخرى بديلة عن الواجهات الحالية Windows Form التي لا تكون عادة بمظهر جيد، جميعنا نعلم ان التطبيقات الجديدة تحتاج الى طرق جديدة لإظهار المعلومات بدلا من واجهة مستطيلة، يكون هنالك اشكال صور مخططات والوان.



واجهة من نوع WPF

واجهة من نوع Windows Form

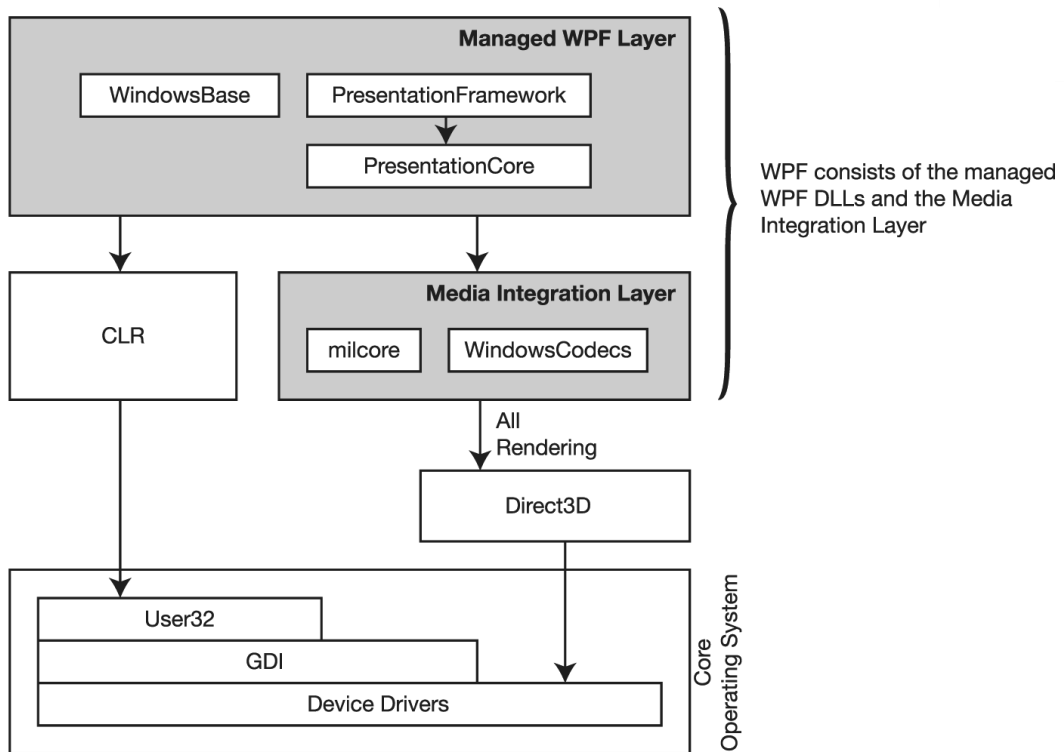
تقنيات الرسوميات الموجودة حاليا تجعل من الصعب خلق تطبيقات مرئية وكذلك حتى الان العتاد الخاص بالرسوميات من كروت الشاشة وحوافظ الذاكرة RAM ما زالت قاصرة حتى عن معالجة بعض تطبيقات Windows Form العادية، يمكنك خلق واجهة من صور متحركة لكن ذلك يحتاج الى جهد كبير وامتلاكنا مجموعة من الأدوات الخاصة لذلك، واستخدام رسوميات ثلاثية البعد 3D تعد مستحيلة بدون استخدام مكتبات إضافية حيث ان مكتبات الـ GDI الأساسية المدعومة من Windows ما زالت ضعيفة.



مخطط توضيحي لهيكلية بناء نظام Windows Form

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



مخطط توضيحي لهيكلية بناء نظام WPF

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

مزايا WPF او Windows Presentation Foundation

تزود WPF بيئة جديدة تأخذ بعين الاعتبار قدرة العتاد الخاص بمعالجة الرسوميات وتزود أرضية لنظام رسومي ثابت وقوي وفعال.

من اهم ما قامت به مزايا WPF:

- استفادت من قدرات الجهاز في معالجة الرسومات مهما كانت ضعيفة باستخدام تقنية تسمى مكدهس الرسوم Graphics Stack.
- جعلت مسألة الدقة مستقلة عن العتاد.
- توحيد التصاميم لـ واجهات المستخدم – المستندات – الوسائط بإنشاء قوالب جاهزة.
- تسمح بعزل البرمجة عن التصميم.
- تتضمن قدرات رسومية عالية من 2D و 3D .
- تسمح بالتعديل على الصور داخل البرمجية او حتى بطرق برمجية ،كما تقدم ترميزات ووسائط مختلفة Codecs.
- تدعم الفيديوهات والصوت.
- توفر الكثير من الوقت والجهد فيما لو اردنا استخدام Windows Form لإظهار نفس النتائج.

يجب هنا ان ننوه ان استخدام WPF لا يعني عن استخدام المحرك الرسومي Direct X 3D يمكن استخدام الالثنين سويًا لتحصل على معالجة رسومية اكبر.

كما أن محرك الرسوميات Flash و SilverLight مازالت مناسبة اكثر لصفحات الويب رغم ان الـ SilverLight هو مجموعة صغيرة من توابع WPF.

تنويه

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



مقارنة بين WPF vs SilverLight

SilverLight	WPF
تستخدم لغة XAML في التصميم	تستخدم لغة XAML في التصميم
مجموعة جزئية من WPF	مجموعة تصميم كاملة
تعمل على متصفحات Firefox ,Internet Explorer ,Safari	تعمل على تطبيقات ويندوز وكل المتصفحات التي تستخدم تقنية XBAP
تعمل على اغلب الانظمة	تعمل على تطبيقات ويندوز فقط
ليست ضمن الـ .NET.	أساسية ضمن الـ .NET.

الآن ما الذي يجب علي استخدامه، هذا يعتمد على الحاجة، الجمهور الهدف، او مكان توزيع البرمجية سواء على النت او تطبيق مكتبي.

غالباً المطورين او المبرمجين ينشئون واجهات ضعيفة او بشعة، وهنا عزلت WPF المبرمجين عن المصممين ليكون كل منهم له عمله الخاص كما يحصل مع صفحات الويب

تنويه

لا بد من القول ان WPF هي بلغة التصميم XAML التي تعتمد على قواعد لغة XML، حيث تتكون WPF من مجموعة من الأصناف.

قوالب WPF في الـ VISUAL STUDIO:

- WPF APPLICATION.
- WPF BROWSER APPLICATION.
- WPF CUSTOM CONTROL LIBRARY (تسمح باستخدام الادوات)
- WPF USER CONTROL LIBRARY (لإنشاء عناصر جديدة)

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



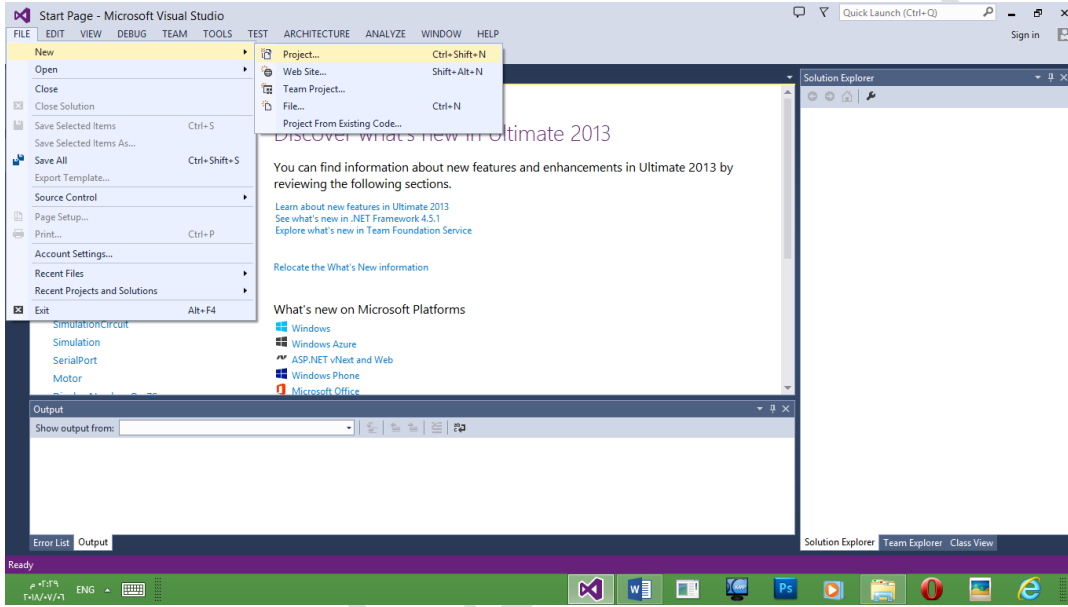
مشروعى الأول فى WPF

فى البداية يجب ان نكون قد نصبنا Microsoft Visual Studio 2013 مع برنامج الـ Blend .
يمكنك معرفة متابعة طريقة التنصيب على قناة StackOverflow Arabi .

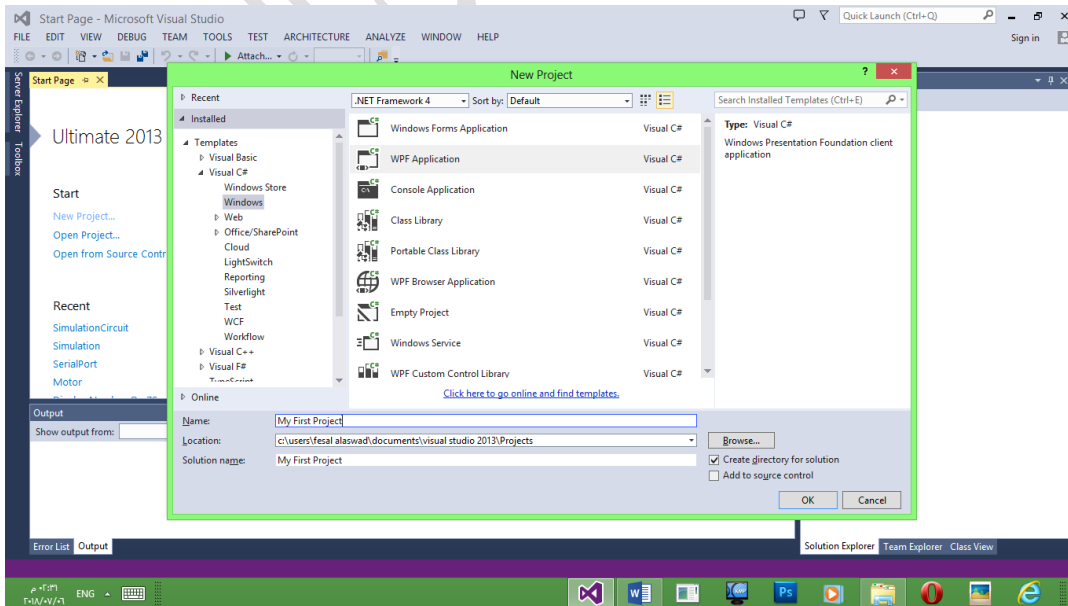
لانشاء مشروع جديد نبدء:

١. فتح مايكروسوفت فيجوال ستوديو.

٢. من قائمة جديد نختار مشروع جديد.



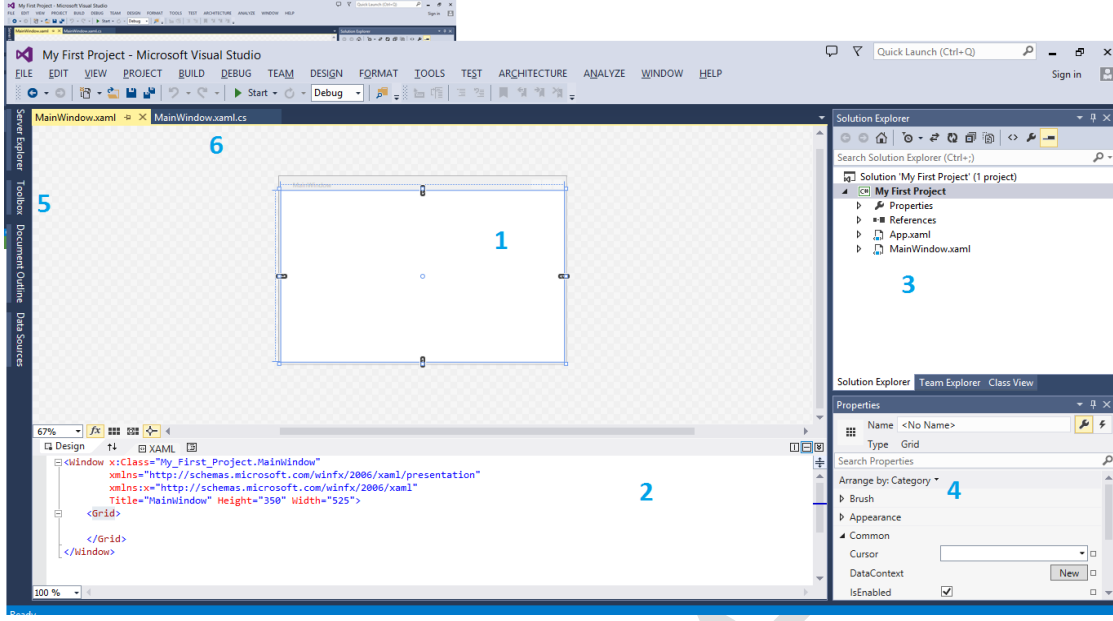
٣. نختار مشروع WPF APPLICATION مع الاسم المراد ثم موافق.



تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube





1. الواجهة الأساسية في التصميم.
2. ترميز الواجهة باستخدام لغة XAML.
3. ملفات المشروع الأساسية.
4. واجهة الخصائص لضبط خصائص كل عنصر.
5. قائمة الأدوات التي تضاف للواجهة (ازرار – مربعات نصوص)
6. تبويب ملف البرمجة الخاص بالواجهة.

المكونات الرئيسية للبرنامج:

```
<Window x:Class="My_First_Project.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="350" Width="525">
    <Grid>
        ..... نكتب هنا كل شيء.....
    </Grid>
</Window>
```

التعريفات في البداية هي أمور تخص مجال الأسماء لـ WPF ان اردت قراءة المزيد عنها
يمكنك الاطلاع على الرابط :

<https://docs.microsoft.com/en-us/dotnet/framework/wpf/advanced/xaml-namespaces-and-namespace-mapping-for-wpf-xaml>

تنويه

برنامجنا الأول

لدينا برنامج يحوي زر له مقاسات مختلفة ومحاذاة مختلفة، (كل شيء يوضع ضمن الـ Grid)

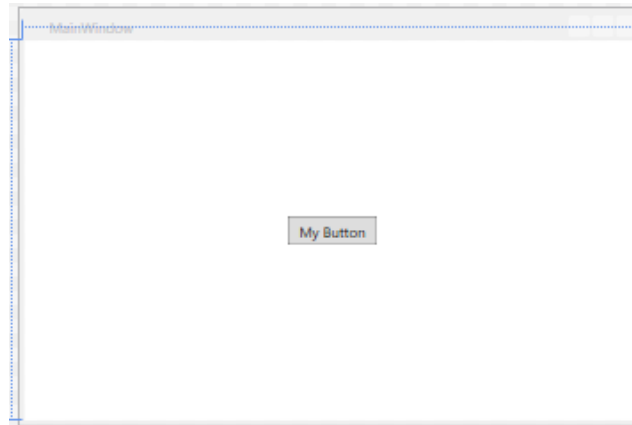
```
<Window x:Class="My_First_Project.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="350" Width="525">
    <Grid>
        <Button
            Content="My Button" الزر اسم
            Height="23" الزر ارتفاع
            Width="75" الزر عرض
            HorizontalAlignment="Center" الأفقية الزر محاذاة
            VerticalAlignment="Center" العمودية الزر محاذاة
            Name="btn" البرمجي الاسم
        >
    </Button>
    </Grid>
</Window>
```

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



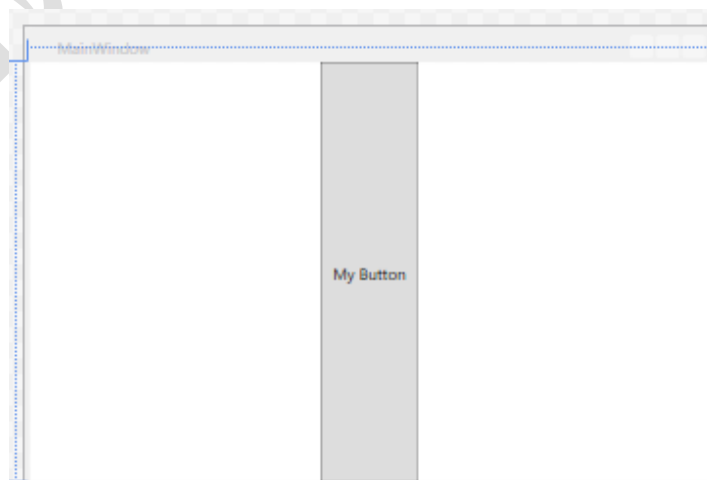
الطلب الاول إضافة زر بطول 23 وعرض ٧٥ ومحاذاة الجوانب الافقية والعمودية من المركز Center وله اسم برمجي btn.



الآن ماذا لو حذفنا الخاصية `height = "75"` كما يلي:

```
<Button
  Content="My Button"
  Width="75"
  HorizontalAlignment="Center"
  VerticalAlignment="Center"
  Name="btn"
  >
</Button>
```

سوف يبقى الشكل كما كان وذلك لان المحاذاة العمودية مختارة باتجاه الوسط، اما لو حذفنا الارتفاع `height = "75"` والمحاذاة العمودية `VerticalAlignment="Center"` فسنرى ان الزر يتوسع ليأخذ ارتفاع الواجهة بالكامل وذلك بسبب عدم اعطائه ارتفاع ومحاذاة معينة، سيكون كما في الشكل:



تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

الامر نفسه فيما لو حذفنا العرض "75" Width="75" والمحاذاة الأفقية نحو الوسط HorizontalAlignment="Center" ستوسع الزر أفقياً ليملاً الشاشة بالكامل حينها.

الطلب الثاني برمجة الزر بحيث يظهر رسالة عند الضغط عليه .
يجب هنا ربط الزر مع حدث برمجي يتم ذلك بإضافة الخاصية **Click** التي تعبر عن حدث الضغط على الزر ، ومن ثم ربطها مع حدث برمجي ما بعد إضافة ذلك الحدث البرمجي ضمن الكود.

```
<Button
  Content="My Button"
  Width="75"
  Height="25"
  Click="Test"
  Name="btn"
>
</Button>
```

هذا الخاصية تمثل اسم الحدث المنفذ عن الضغط على الزر وهنا هو الحدث Test المضاف الى الكود والذي يظهر عبارة ترحيب على الشاشة.

```
public void Test(object sender,RoutedEventArgs e)
{
  MessageBox.Show("Hello in my Channel StackOverFlow Arabi In Youtube");
}
```

- البارامترات القادمة من الحدث تقدم معلومات مهمة حول هذا الحدث ومصدره ، ومنها:
- الكائن Sender يعبر عن كائن مسبب الحدث (وهنا الزر الذي انشأ حدث الرسالة).
 - الكائن e يحوي العديد من المعلومات الإضافية غير الموجودة في Windows Form .

عند نسخ زر من زر لا ينسخ سوى خصائصه دون الاحداث الناتجة عن التعامل معه ، كما اوردنا فقط خصائص هذا الزر تنسخ الى الزر الجديد ويظهر لدينا زران متماثلان في الخصائص والمظهر ومستقلان في الاحداث.

تنويه

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



الخاصية `ButtonBase.Click="Test"` الموضوعة في خصائص الواجهة تحدد ان كل الازرار داخل هذه الواجهة يجب ان تنفذ هذا الحدث وكأنه حدث مشترك للكل.

```
<Window x:Class="My_First_Project.MainWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="MainWindow" Height="350" Width="525" ButtonBase.Click="Test">
```

المهندس فيصل الأسود

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



المهندسين فيفضل الأسود

فهم الـ XAM

تأليف فيصل الأسود | مهندس برمجيات
يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



فهم الـ XAML:

XAML بسيطة و تمنحك عدة طرق مختلفة لتعطي قيم للخاصية ، حيث انه يوجد أنواع للخصائص منها البسيطة ومنها المعقدة ومنها يحوي داخليا محولات أنواع ، فهم الية XAML سيكون موضوع هذه الفقرة.

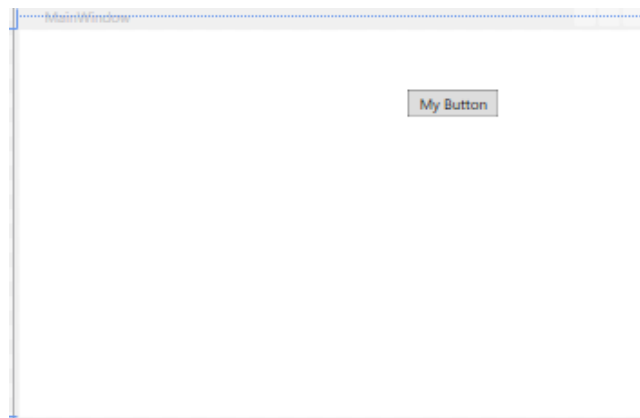
عند ادراج زر في الواجهة مباشرة من صندوق الأدوات نلاحظ انه سوف يتم توليد العديد من خصائصه مباشرة في كود الـ XAML الموافق للواجهة، من هذه الخصائص الطول والعرض والمحاذاة الافقية والعامودية.

بالنظر بتمعن الى تلك الخصائص نجد ان بعضها سلاسل نصية بسيطة مثل خاصية الاسم الظاهر `Content="My Button"` او الاسم البرمجي `Name="btn"` وهناك منها اكثر تعقيداً، نلاحظ أيضاً ان الطول والعرض عبارة عن سلاسل رقمية حصراً `Width="75" Height="23"` حيث انها تحول داخلياً الى ارقام .

الخصائص المعقدة اكثر مثل المحاذاة الافقية والعامودية والتي تأخذ قيم محسوبة او مجهزة مسبقاً مثل (`Bottom ,Center, stretch , Top , Left ,Right`)
`HorizontalAlignment="Center"`

او الخاصية `Margin` والتي يمكن ان تأخذ عدة اشكال او حالات من المدخلات هي:

- تأخذ قيمة وحيدة (تكون لكل الاتجاهات).
`Margin="25"`
- تأخذ قيمتين (يمين /يسار او اعلى / اسفل).
`Margin="25,10"`
- تأخذ اربع قيم (يمين - اسفل - يسار - اعلى).
`Margin="200,2,2,200"`



`Margin="200,2,2,200"`

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

تعلمنا سابقاً طريقة كتابة الخصائص داخل كل عنصر او داخل اقواس الـ TAG كما يلي :

```
<Button هنا الخصائص >
</Button>
```

مثال

```
<Button
  Content="My Button" Height="23" Width="75"
  Margin="200,2,2,200" Name="btn" >
</Button>
```

لكن هناك طريقة أخرى لكتابة الخصائص هي ان ننشئ Tag او وسم لكل خاصية على حدة داخل بداية ونهاية العنصر كما يلي:

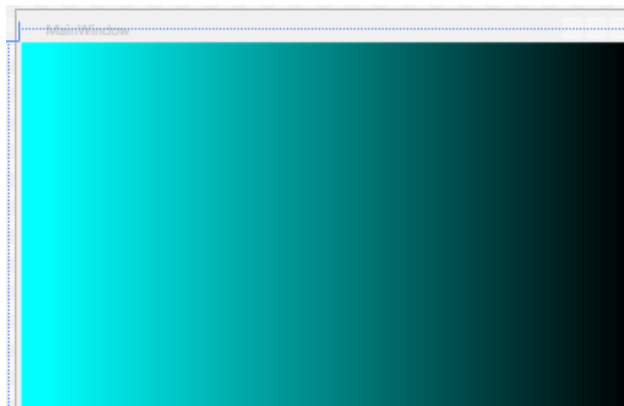
```
<Button >
  هنا نكتب الخصائص داخل وسوم خاصة بها
</Button>
```

مثال

```
<Button >
  <Button.Content>my button</Button.Content>
</Button>
```

تدريب :

لدينا واجهة تحوي شبكة Grid المطلوب تعيين خلفية الشبكة كتدرج خطي للونين كما في الشكل:



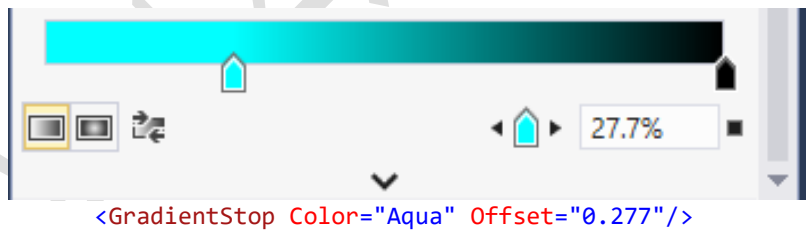
تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

هنا عبارة عن خصائص معقدة ومتداخلة حيث الـ Grid تحوي بداخلها خاصية الخلفية وكذلك الامر الخلفية تحوي خصائص التدرج وهذا ما يسمى بمفهوم التداخل.

```
<Grid >
  <Grid.Background >
    <LinearGradientBrush StartPoint="0,0.5" EndPoint="1,0.5">
      <GradientStop Color="Aqua" Offset="0.277"/>
      <GradientStop Color="Black" Offset="1"/>
    </LinearGradientBrush>
  </Grid.Background>
</Grid>
```

١. الخاصية <Grid.Background > تمثل خلفية الشبكة للواجهة.
٢. الخاصية <LinearGradientBrush StartPoint="0,0.5" EndPoint="1,0.5"> تمثل تدرج خطي على شكل مستقيم يبدأ من <StartPoint="0,0.5"> وينتهي بـ <EndPoint="1,0.5"> .
٣. الخاصية <GradientStop Color="Aqua" Offset="0.04"/> تمثل احد الوان التدرج والـ Offset مقدار يعبر عن مكان بداية تداخل اللون، كما في الأداة:



كما يجب ان نذكر ان الخلفية للشبكة تحوي عدة أنواع للون هي:

- No Brush وهي لا تمنح لون للخلفية.
- Solid Color Brush تمنح لون واحد .
- Gradient brush تمنح تدرج من عدة الوان.
- Tile Brush تمنح القدرة على وضع صورة خلفية.
- Brush resource تمنح القدرة على استخدام أنماط جاهزة مسبقاً.

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

ولكن كيف نعرف ان نضبط هذه الخصائص، هذا ما سيعتمد عليه تعلمك وتجربتك لتلك الخصائص من خلال التدريب ، بالإضافة الى ما سنقدمه لك من تدريبات ومقاطع فيديو في قناتنا او هذا الكتاب.

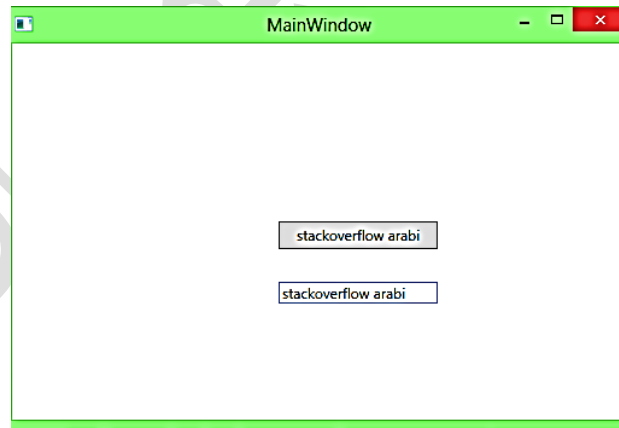
الخصائص اثناء التنفيذ:

في بعض الأحيان لا يجب ان نضع الخصائص في مرحلة التصميم انما عند التنفيذ وهنا نستخدم طريقة جديدة لتمثيل هذه الخصائص بحيث تكون محاطة بـ اقواس {} وتعني ان هذه الخصائص تنفذ عند التنفيذ.

الشكل العام لكتابة هذه الخصائص كما يلي:

```
Content="{Binding ElementName=OBJECT,Path=PROBRTY}"
```

ليكن لدينا المثال التالي الذي يعبر عن وجود زر ومربع نص داخل الواجهة ، الزر دوما يأخذ اسمه من مربع النص ، أي انه وعند التنفيذ كلما غيرنا النص داخل المربع يتغير الاسم الظاهري للزر الشكل التالي يوضح



الكود التالي يعبر عن عملية الربط عند التنفيذ

```
<Grid >
<Button Height="25" Width="75" Content="{Binding ElementName=txt,Path=Text}">
</Button>
<TextBox Height="25" Width="75" Margin="0,100,0,0" Text="My Text" Name="txt">
</TextBox>
</Grid>
```

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



نلاحظ ان قيمة الاسم **Content** للزر مرتبطة مع مربع النص txt وبالتحديد خاصية النص Text لمربع النص.

وبهذا نكون قد اعلمنا المترجم للكود ان الاسم الظاهري للزر يأخذ من خاصية النص للعنصر txt الذي هو مربع نص ، حيث دوماً في عملية ربط عنصرين نحتاج اسم البرمجي للعنصر والخاصية الهدف المطلوبة منه.

يمكن كما اوردنا سابقا كتابة الخصائص بطريقة ثانية ومنه لكتابة كود الربط السابق نكتب :

```
<Button Margin="221,147,156,141" >
  <Button.Content>
    <Binding ElementName="txt" Path="Text"></Binding>
  </Button.Content>
</Button>
<TextBox Margin="221,197,156,96" Text="My Text" Name="txt"/>
```

تنويه

من الأشياء الضرورية التي يجب ان ننوه عليها ان الخصائص حساسة لحالة الاحرف لذلك يجب ان ننتبه ان معظم الخصائص تبدأ بحرف كبير

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



العناصر الحاوية Containers Controls

المهندس فيصل الأسود
الكتاب على قناة الـ Youtube

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



العناصر الحاوية ContainerControls:

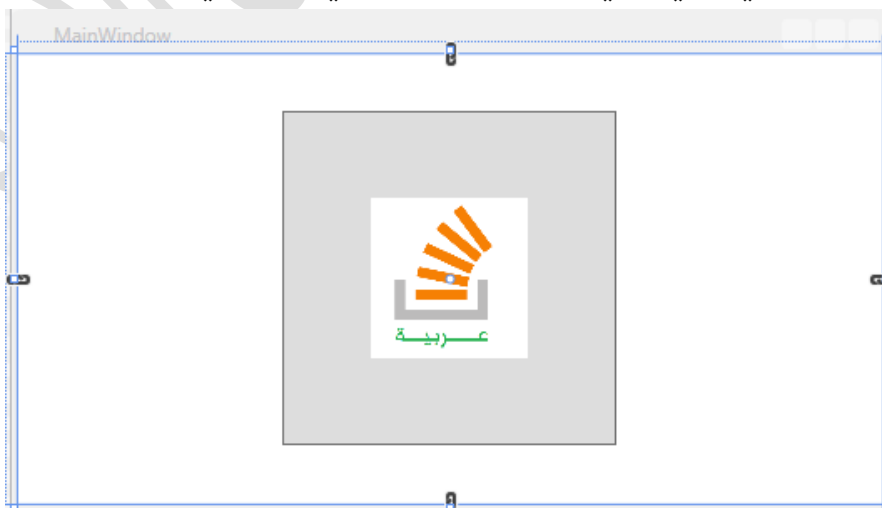
العناصر الحاوية هي العناصر التي تحوي بداخلها عناصر أخرى في السابق باستخدام Windows Forms كانت هناك القليل من العناصر التي تحوي عناصر بداخلها احداها ال Form الذي يحوي ازرار ومربعات نص ، لكن المفاجئ في تقنية الـ WPF وما يعطيا المزيد من القوة و قدرة العناصر في احتوائها على عناصر أخرى. علينا ان نعلم أولا ان كل العناصر ترث من العناصر الحاوية حتى الزر يمكن ان يحوي صورة او نص والقائمة تحوي مربعات نص وصور. العناصر الحاوية ومناقشتها سيكون موضوعنا في هذا الدرس والذي سيعكس بالفعل قوة تقنية WPF في التصميم.

في البداية سوف نتطرق الى أنواع حاويات العناصر الأساسية وهي كما سترد معنا:

- WrapPanel
- DockPanel
- StackPanel والتي تكدر العناصر افقيا او عاموديا وهي مفيدة لتطبيق الازرار.
- Grid والتي ترتب العناصر ضمن شبكة ولا تحتاج قياس او مكان تأخذ حجم الفورم مباشرة هي وعناصرها.
- Canvas في هذا النوع يجب ضبط يدويا مكان كل عنصر.

سوف يتم شرح كل من هذه العناصر بالتفصيل مع الأمثلة.

ليكن لدينا المثال التالي والذي يحوي زر بداخله صورة كما في الشكل في الأدنى:



تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

تنويه

لن تظهر الصورة داخل الزر في حال إعطائه خاصية المحتوى Content لذلك يجب ازلتها.

```
<Window x:Class="My_First_Project.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="300" Width="525">
    <Grid >

        <Button

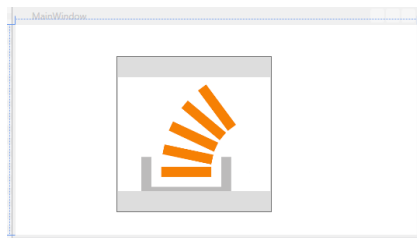
            Name="btn"
            VerticalAlignment="Center"
            HorizontalAlignment="Center"
            Width="200" Height="200">

            <Image Source="E:\stack.png" Stretch="Uniform" Margin="50" />
        </Button>

    </Grid>
</Window>
```

نلاحظ في المثال السابق ان الصورة داخل وسوم البداية والنهاية للزر ، وهذا يعني ان الصورة داخل الزر واينما تحرك الزر سوف تتحرك معه ، بالإضافة لذلك نرى الخاصية **Stretch** التي تبين طريقة ظهور الصورة داخل العنصر الذي يحتويها ، فيما يلي شرح لانواع اظهار الصور داخل العناصر وحالات الخاصية **Stretch** :

- Fill تحاول ملأ العنصر بالصورة قدر الإمكان.
- None تترك الصورة حسب مقاسها الأصلي.
- Uniform تقيس الصورة بحيث تقع في مكان من الصورة وتبقى ابعادها متناسبة.
- UniformToFill تملأ الصورة بحيث تبقى ابعادها متناسبة ، أي احد ابعادها الأكبر يأخذ البعد الأكبر للصورة والبعد الاخر يتناسب معه.



UniformToFill

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

الآن لننظم محتويات الزر حسب الحاويات الرئيسية التي ذكرناها سابقاً ولنضيف داخل الزر صورة ومربع نص و نرتبهم عامودياً فوق بعضهم حسب الـ StackPanel كما في الشكل التالي:



يمكن تحقيق ذلك ببساطة عن طريق كتابة الكود التالي:

```
<Window x:Class="My_First_Project.MainWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="MainWindow" Height="300" Width="525" >
  <Grid >
    <Button
      Name="btn"
      VerticalAlignment="Center"
      HorizontalAlignment="Center"
      Width="200" Height="200" Margin="130,40,187,29"
    >
      <StackPanel Orientation="Vertical">
        <Image Source="E:\stack.png" Stretch="UniformToFill"
        Height="146"/>
        <TextBox Text="stackOverflow" TextAlignment="Center"></TextBox>
      </StackPanel>
    </Button>
  </Grid>
</Window>
```

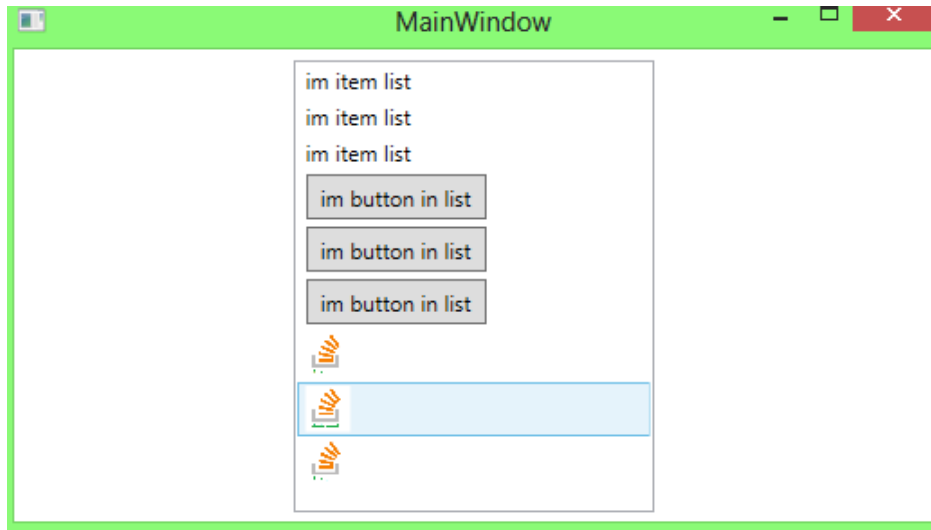
والذي يحوي زر يحوي بداخله StackPanel تنظم ترتيب الزر والصورة فوق بعضهم، الآن ماذا لو اردنا ترتيب العناصر بجانب بعض ، الحل بسيط فقط نستبدل الخاصية **Orientation** في ال StackPanel الى **Horizontal** التي ترتب العناصر افقياً.

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



ليكن لدينا مثال اخر الذي يعبر عن واجهة تحوي قائمة وبداخل هذه القائمة ازرار ومربعات نص وعناصر كما في الشكل التالي:



الكود الترميزي المكافئ للواجهة السابقة كما يلي:

```
<Window x:Class="My_First_Project.MainWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="MainWindow" Height="300" Width="525" >
  <Grid >

    <ListBox Width="200" Height="250">
      <ListBoxItem Content="im item list"></ListBoxItem>
      <ListBoxItem Content="im item list"></ListBoxItem>
      <ListBoxItem Content="im item list"></ListBoxItem>
      <Button Content="im button in list" Width="100" Height="25"></Button>
      <Button Content="im button in list" Width="100" Height="25"></Button>
      <Button Content="im button in list" Width="100" Height="25"></Button>
      <Image Source="E:\stack.png" Width="25"></Image>
      <Image Source="E:\stack.png" Width="25"></Image>
      <Image Source="E:\stack.png" Width="25"></Image>
    </ListBox>

  </Grid>
</Window>
```

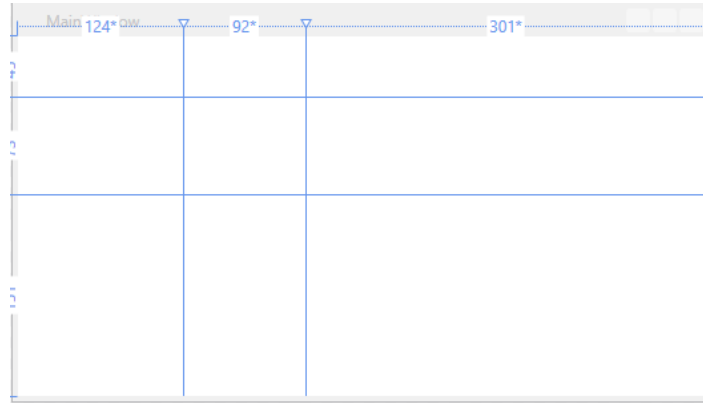
تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



التعامل مع الـ Grid:

يمكن تقسيم الواجهة عن طريق ما يسمى بالأسطر والاعمدة والتي تساعد على تنظيم ترتيب الواجهة كما في الشكل:



هذه الالاسطر والاعمدة لن تكون ظاهرة اثناء التنفيذ الا انها تساعد وبشكل كبير في ترتيب العناصر داخل الواجهة ، يمكن تنفيذ تلك التقسيمات باستخدام الكود التالي:

```
<Window x:Class="My_First_Project.MainWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="MainWindow" Height="300" Width="525" >
  <Grid >
    <Grid.ColumnDefinitions>
      <ColumnDefinition Width="124*" />
      <ColumnDefinition Width="92*" />
      <ColumnDefinition Width="301*" />
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
      <RowDefinition Height="45*" />
      <RowDefinition Height="73*" />
      <RowDefinition Height="151*" />
    </Grid.RowDefinitions>
  </Grid>
</Window>
```

ولوضع عناصر داخل تلك التقسيمات نكتب الكود التالي الذي يكافئ وضع زر داخل التقسيم:

تأليف فيصل الأسود | مهندس برمجيات

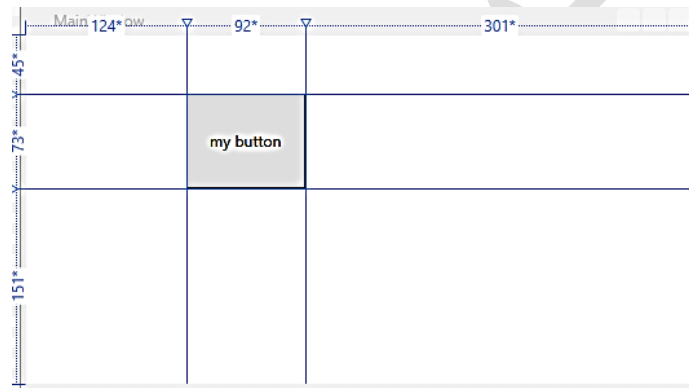
يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

```

<Window x:Class="My_First_Project.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="300" Width="525" >
    <Grid >
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="124*" />
            <ColumnDefinition Width="92*" />
            <ColumnDefinition Width="301*" />
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="45*" />
            <RowDefinition Height="73*" />
            <RowDefinition Height="151*" />
        </Grid.RowDefinitions>
        <Button Content="my button" Grid.Row="1" Grid.Column="1" ></Button>
    </Grid>
</Window>

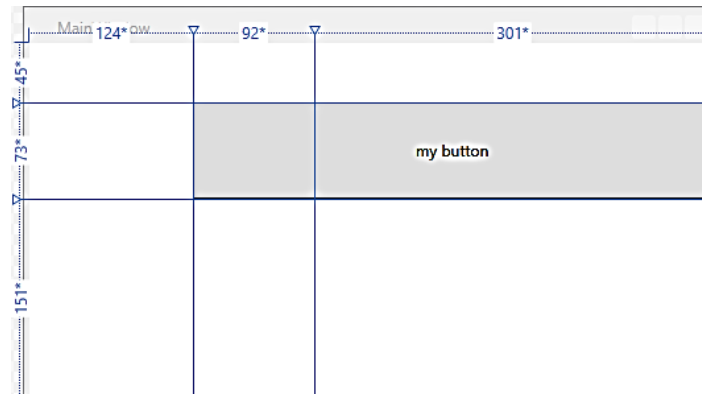
```

ويكون الشكل الناتج كما في الصورة التالية:



وهذا يعني إضافة زر في العمود الثاني والسطر الثاني `Grid.Row="1" Grid.Column="1"` حيث يبدأ العد من الصفر، يمكن تمديد الزر على الاعمدة التي تليه بإضافة التعليمة `Grid.ColumnSpan="2"` حيث العدد 2 يمثل عدد الاعمدة التي يمتد عليها.

كما يوجد تعليمة للتمديد على الاسطر `Grid.RowSpan="2"`



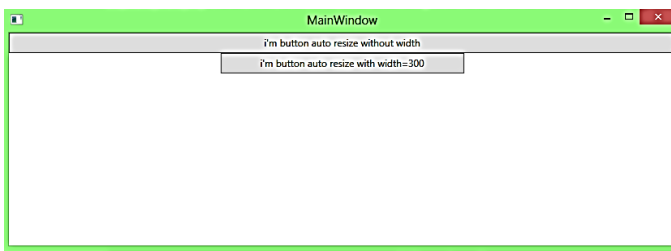
تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

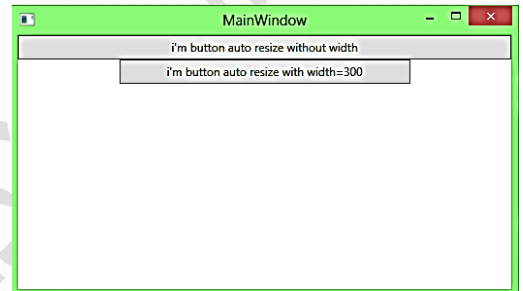
العناصر لها احداثيات ثابتة او موقع يملك إحداثيات الـ x و y والتي تمثل مكان زاوية
العنصر العليا اليسرى
بالنسبة للمركز الاحداثيات الواقع في الزاوية اليسرى العليا

تنويه

ما يميز الـ WPF ان الدقة والقياس مستقل تماما عن الواجهة حيث انه يتغير تلقائيا اذا تم تقييس
الواجهة يدويا.



بعد التمديد



قبل التمديد

نلاحظ ان الـ زر الذي يملك عرض محدد بقي على وضعه عند تمديد النافذة، اما الـ زر الاخر الذي لا
يملك عرض فهو يأخذ عرض الحاوية StackPanel ويتمدد بتمددتها، الكود التالي يمثل التصميم:

```
<Window x:Class="My_First_Project.MainWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="MainWindow" Height="300" Width="525" >
  <Grid >
    <StackPanel>
      <Button Height="25" Content="i'm button auto resize without width"></Button>
      <Button Height="25" Width="300" Content="i'm button auto resize with
width=300">
</Button>

    </StackPanel>
  </Grid>
</Window>
```

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



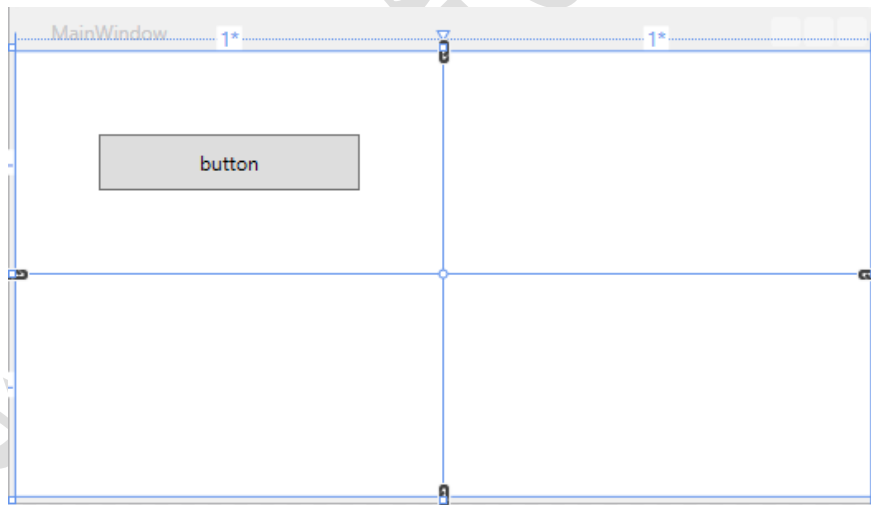
هنالك ثلاث طرق لتقييس الاعمدة والاسطر في النمط Grid وهي:

١. التقييس التلقائي وهنا عندما لا نعطي مواضع للاعمدة والاسطر ، ومنه تأخذ توزيع تلقائي للابعاد ويتحقق ذلك بعدم إعطاء قيم للارتفاع او العرض او باعطاء القيمة `Height="Auto"` المثال التالي سيوضح كيفية تقييس الاعمدة والاسطر تلقائيا وبنوع تلقائي.

```
<Window x:Class="My_First_Project.MainWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="MainWindow" Height="300" Width="525" >

  <Grid>
    <Grid.RowDefinitions>
      <RowDefinition > </RowDefinition>
      <RowDefinition ></RowDefinition>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
      <ColumnDefinition></ColumnDefinition>
      <ColumnDefinition></ColumnDefinition>
    </Grid.ColumnDefinitions>
    <Button Content="button" Margin="50"></Button>
  </Grid>

</Window>
```

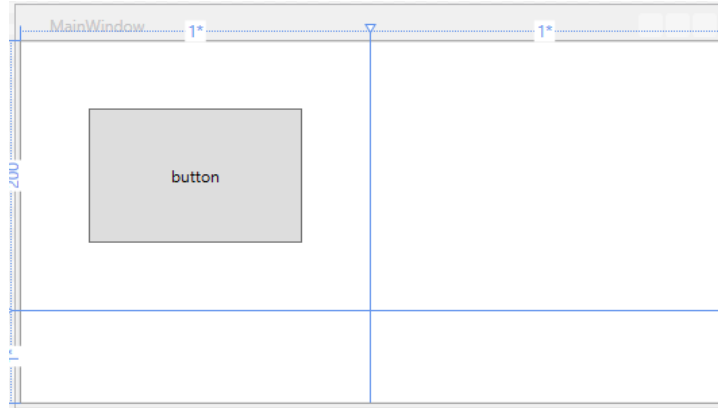


تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



٢. التقييس الصريح Absolute وهنا نعطي مواضع للاعمدة والاسطر ، ومنه تأخذ توزيع حسب الابعاد المعطاة ، كما في الشكل.

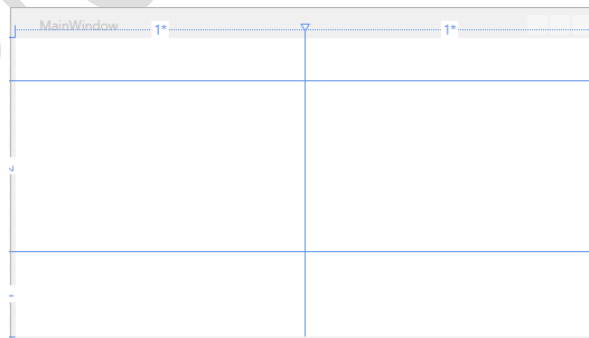


٣. التقييس النسبي Absolute وهنا نعطي نسبة توزع للاعمدة والاسطر ضمن الواجهة ، ومنه تأخذ توزيع حسب النسب المعطاة ، يعتبر هذا النوع الأفضل في تصور الواجهة.

```
<RowDefinition Height="0.5*"> </RowDefinition>
<RowDefinition Height="2*"></RowDefinition>
<RowDefinition Height="*"></RowDefinition>
```

بالبدية يجب ان ننوه ان الـ * تدل على التقييس النسبي.

وهنا نلاحظ ان أعمدة الواجهة تقسم على مجموع نسب الاعمدة والذي هو ٣,٥ وبالتالي يتم توزيع ارتفاع الواجهة وكانه من ٣,٥ ويأخذ كل عامود مكانه . الصورة بالاسفل تقابل الكود السابق.



ملاحظة هامة:

من المهم ان نعلم انه في حال تقييس الاسطر او الاعمدة تقييس صريح فانها تحافظ على قيمتها الثابتة عند تمديد الواجهة ، اما في حال التقييس النسبي فانها تحافظ على نسبتها بالنسبة لابعاد الواجهة.

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

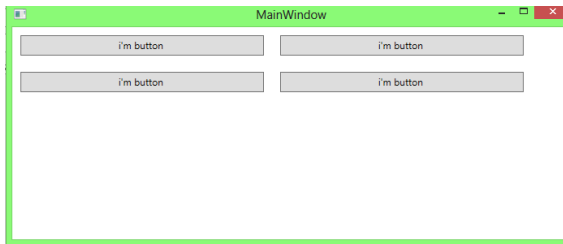
التعامل مع الـ Canvas:

العناصر في هذا النوع لها احداثيات حرة كما في الـ Windows Form ، كما انها لا تتقيس عند تمديد الواجهة بوضع التنفيذ ، ولا تملك دقة عناصر مستقلة. يمكن توزيع العناصر ضمنها عند طريق الخاصية ، وكما في الجدول:

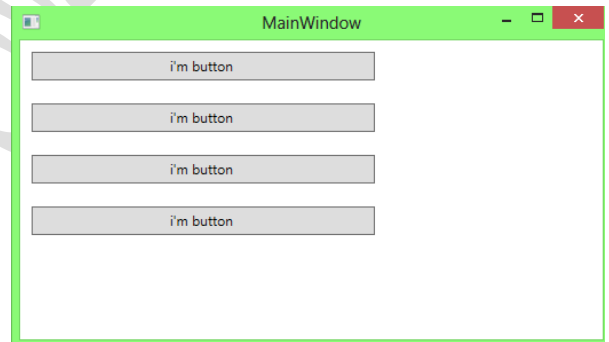
<code>Canvas.Left="22"</code>	<code>Canvas.Right="22"</code>	<code>Canvas.Top="30"</code>	<code>Canvas.Bottom="50"</code>
بعد العنصر عن الحافة اليسرى ٢٢	بعد العنصر عن الحافة اليمنى ٢٢	بعد العنصر عن الحافة العليا ٣٠	بعد العنصر عن الحافة السفلى ٥٠

التعامل مع الـ WrapPanel:

ترتب هذه النوعية من الحاويات العناصر من اليسار لليمين ومن الاعلى للأسفل تلقائياً نلاحظ انه عند تمديد هذه الواجهة العناصر تترتب تلقائياً .



بعد التمديد وإعادة ترتيب العناصر تلقائياً



قبل التمديد

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube


```
<Window x:Class="My_First_Project.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="300" Width="525" >
    <Grid >
        <WrapPanel>
            <Button Height="25" Width="300" Content="i'm button " Margin="10"></Button>
            <Button Height="25" Width="300" Content="i'm button " Margin="10"></Button>
            <Button Height="25" Width="300" Content="i'm button " Margin="10"></Button>
            <Button Height="25" Width="300" Content="i'm button " Margin="10"></Button>
        </WrapPanel>
    </Grid>
</Window>
```

فيصل الأسود

تأليف فيصل الأسود | مهندس برمجيات

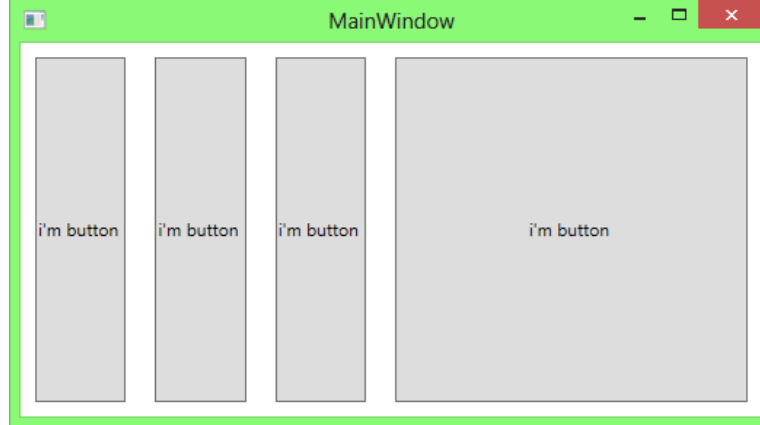
يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



التعامل مع الـ DockPanel :

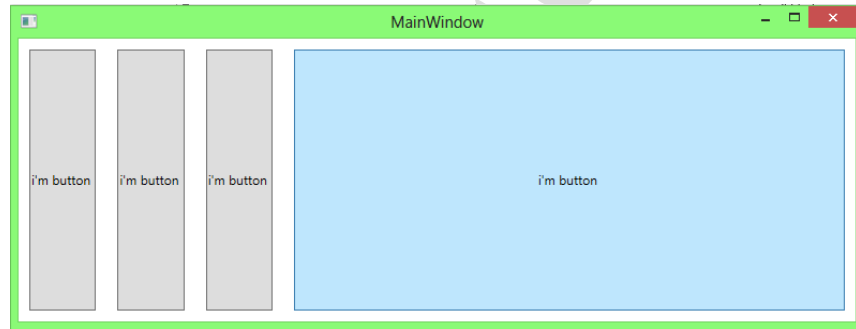
يقوم هذا النوع بتوزيع العناصر قريبة من الحواف للواجهة ،حتى عند تمديد الواجهة تتمدد العناصر لتبقى حوافها ملاصقة لحواف الواجهة ، كما في الشكل التالي:

قبل التمديد



بعد التمديد

"نلاحظ كيف ان
كل العناصر
ملاصقة للحواف
تلقائياً"



```
<Window x:Class="My_First_Project.MainWindow"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
Title="MainWindow" Height="300" Width="525" >

    <DockPanel >
    <Button Content="i'm button " Margin="10"></Button>
    <Button Content="i'm button " Margin="10"></Button>
    <Button Content="i'm button " Margin="10"></Button>
    <Button Content="i'm button " Margin="10"></Button>
    </DockPanel>

</Window>
```

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



من الواجهة السابقة نلاحظ ان اخر عنصر دوماً يتمدد بشكل كبير لبقى ملاصقاً للواجهة ، ماذا لو اردنا ابقائه بمقاسه الطبيعي ،وهنا الحل ان نوقف الخاصية المسؤولة عن التمدد بوضعها كم يلي `LastChildFill="False"` في خواص الحاوية `DockPanel` وهنا لم يعد ليتمدد الزر الأخير.

الان لنفكر قليلاً ما الفرق بين الحاوية `DockPanel`والحاوية `StackPanel` التي تكدر العناصر الجواب هنا ان الحاوية `DockPanel` تسمح لنا بتوزيع العناصر افقياً وعمودياً بآن واحد ، كما في الشكل التالي وذلك باستخدام خاصية اتجاه المحاذاة `DockPanel.Dock` وتعين لها اتجاه للمحاذاة.



```
<Window x:Class="My_First_Project.MainWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="MainWindow" Height="300" Width="525" >

  <DockPanel >
    <Button Content="i'm button " Margin="10" DockPanel.Dock="Left"></Button>
    <Button Content="i'm button " Margin="10" DockPanel.Dock="Top"></Button>
    <Button Content="i'm button " Margin="10" DockPanel.Dock="Right"></Button>
    <Button Content="i'm button " Margin="10" DockPanel.Dock="Bottom"></Button>
  </DockPanel>

</Window>
```

نلاحظ مرة أخرى ان الزر الأخير يتمدد تلقائياً ليملى الفراغ ، كذلك يمكننا حل هذه المسألة بإيقاف الخاصية المسؤولة عن الملئ.

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



تداخل العناصر Nested Controls Container:

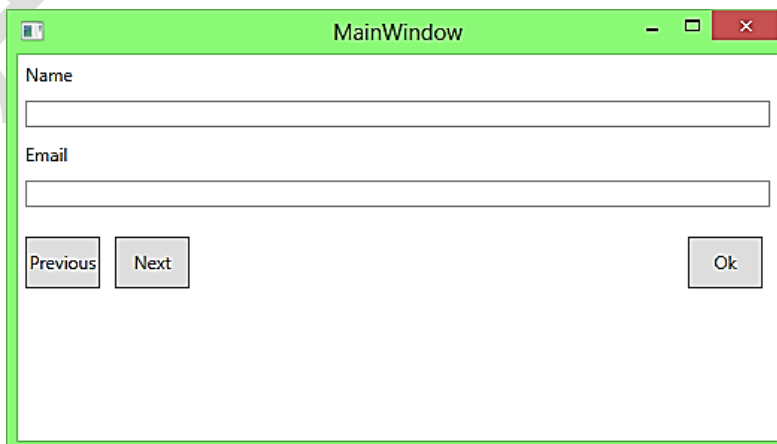
غالباً لا يمكن استخدام احدى حاويات العناصر وحدها ، وهنا نحتاج لاستخدام اكثر من نوع داخل التصميم بحيث تحوي احداها الاخرى
في مثالنا التالي سوف نستعرض واجهة لتسجيل الدخول تحوي ثلاث حاويات StackPanel وجميعهم محتواه ضمن حاوية DockPanel وفق توزيع معين.

```
<Window x:Class="My_First_Project.MainWindow"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
Title="MainWindow" Height="300" Width="525" >

    <DockPanel>
        <StackPanel DockPanel.Dock="Top" Margin="0,0,0,5">
            <TextBlock Text="Name" Margin="5" ></TextBlock>
            <TextBox Margin="5"></TextBox>
            <TextBlock Text="Email" Margin="5" ></TextBlock>
            <TextBox Margin="5"></TextBox>
        </StackPanel>

        <StackPanel DockPanel.Dock="Left" Orientation="Horizontal"
HorizontalAlignment="Left">
            <Button Content="Previous" Margin="5,10,5,5" MinWidth="50" Height="35"
VerticalAlignment="Top" ></Button>
            <Button Content="Next" Margin="5,10,5,5" MinWidth="50" Height="35"
VerticalAlignment="Top"></Button>
        </StackPanel>

        <StackPanel DockPanel.Dock="Right" Orientation="Horizontal"
HorizontalAlignment="Right">
            <Button Content="Ok" Margin="10,10,10,5" MaxWidth="50" Height="35" Width="50"
VerticalAlignment="Top"></Button>
        </StackPanel>
    </DockPanel>
</Window>
```



اهم ما يجب ان نلاحظه انه عند تمديد هذه الواجهة ان مربعات النص تتمدد بتمدد الواجهة.

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



التعامل مع الخصائص برمجياً:

كيف يمكننا التعامل مع العناصر برمجياً ، كيف يمكن ان نغير الخصائص الظاهرية للعناصر عن طريق الكود البرمجي ، لا بد ان نذكر من فهمنا لمبدأ الـ OOP ان جميع الخصائص الخاصة بالواجهة او ب تقنية الـ WPF ان جميع تلك الخصائص للعناصر ترث من الصنف DependencyProperty الذي يسمح لي التحكم بكل خصائص العناصر عن طريق التتابع GetValue لمعرفة قيمة خاصية ما لعنصر، والتابع SetValue الذي يغير قيمة خاصة.

```
btn.GetValue();
```

object DependencyObject.GetValue(DependencyProperty dp)
Returns the current effective value of a dependency property on this instance of a System.Windows.DependencyObject.

Exceptions:
System.InvalidOperationException

من الشكل السابق نلاحظ ان التابع GetValue يقبل أي عنصر من نوع DependencyProperty أي يقبل أي خاصية من خصائص الواجهة.

ليكن لدينا المثال التالي الذي يحوي زر كما في الشكل ١ عند الضغط على الزر ينتقل الى السطر التالي كما في الشكل ٢ وعند الضغط ينتقل الى الشكل ٣ وعند الضغط وهو في الشكل ٣ يعود كما كان في الشكل ١.



الشكل ٣

الشكل ٢

الشكل ١

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

الكود الترميزي لتهيئة الواجهة هو كالتالي:

```
<Grid >
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="124*" />
    <ColumnDefinition Width="92*" />
    <ColumnDefinition Width="301*" />
  </Grid.ColumnDefinitions>
  <Grid.RowDefinitions>
    <RowDefinition Height="45*" />
    <RowDefinition Height="73*" />
    <RowDefinition Height="151*" />
  </Grid.RowDefinitions>
  <Button Content="my button" Click="btn_Click_1" Grid.Row="0"
Grid.Column="1" Name="btn" ></Button>
</Grid>
```

```
private void btn_Click_1(object sender, RoutedEventArgs e)
{
    int rowNum=(int)btn.GetValue(Grid.RowProperty);
    if(rowNum < 2)
    {
        btn.SetValue(Grid.RowProperty,rowNum+1);
    }
    else
    {
        btn.SetValue(Grid.RowProperty, 0);
    }
}
```

هنا اخذنا قيمة الخاصية رقم السطر وتم وضعه في المتحول rowNum ومناقشة طالما لم يصل
للنهاية أي السطر الحالي اقل من ٢ أي ليس السطر الثالث ومنه غير القيمة الى السطر 1+rowNum
ماعدًا ذلك أي ولنا للسطر الثالث ومنه اعده للسطر الأول.

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



التعامل مع العناصر UI Controls

المهندس فيصل الأسود
فيصل الأسود

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

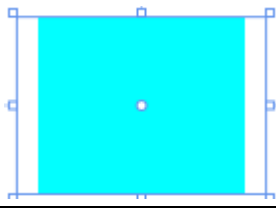
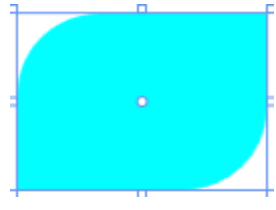
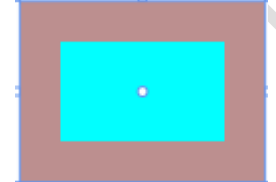


التعامل مع العناصر:

في هذا القسم سوف نستعرض خصائص اغلب الأدوات الشائعة ، ومنها سنجد اشكال الخصائص المختلفة كما تحدثنا في الأقسام الأولى.

التعامل مع Border:

Border او اللطار هو أداة يمكن ان تحوي بداخلها عناصر أخرى مثل الازرار يستخدم اللطار بشكل رئيسي لكي يرسم حواف العناصر الذي يحتويها ، وهنا نجد قوة WPF في رسم العناصر.

النتيجة	مثال	اشكال الخاصية
	<code>BorderThickness="10,0,10,0"</code>	<code>BorderThickness="All"</code> <code>BorderThickness="R L,T B"</code> <code>BorderThickness="R,B,L,T"</code>
	<code>CornerRadius="40,0,40,0"</code>	<code>CornerRadius ="All"</code> <code>CornerRadius ="R L,T B"</code> <code>CornerRadius ="R,B,L,T"</code>
	<code>BorderBrush="RosyBrown"</code>	<code>BorderBrush="Color"</code>

كما يمكن التعامل مع خصائص التلوين سواء اللون الواحد او المتدرج.

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

التعامل مع Button:

الجدول التالي يحوي اهم خصائص الازرار.

النتيجة	مثال	اشكال الخاصية
	<code>Content="im button"</code>	<code>Content="Text"</code>
	<code>FontSize="20"</code>	<code>FontSize="Value"</code>
	<code>FontWeight="Bold"</code>	<code>FontWeight="Type"</code>
سوف يتغير مكان العنصر على الواجهة	<code>VerticalAlignment="Center "</code>	<code>HorizontalAlignment="Position"</code> <code>VerticalAlignment=" Position "</code>
<code>ClickMode="Press"</code> أي سيتم تنفيذ الامر عند الضغط. <code>ClickMode=" Hover"</code> أي سيتم تنفيذ الامر عند المرور. <code>ClickMode=" Release"</code> أي سيتم تنفيذ الامر بعد الضغط ورفع الماوس أي عند تحرير الزر من الضغط.	<code>ClickMode="Press"</code> <code>ClickMode=" Hover"</code> <code>ClickMode="Release"</code>	<code>ClickMode="Type"</code>
	<code>Background="Aquamarine"</code>	<code>Background="Color"</code>
	<code>Foreground="Green"</code>	<code>Foreground="Color"</code>
	<code>BorderBrush="DarkRed"</code>	<code>BorderBrush="Color"</code>
 غير قابل للضغط	<code>IsEnabled="false"</code>	<code>IsEnabled="boolValue"</code>

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

التعامل مع CheckBox:

النتيجة	مثال	اشكال الخاصية
<input checked="" type="checkbox"/> check box	<code>IsChecked = "True"</code>	<code>IsChecked="Value"</code>
<input type="checkbox"/> check box يأخذ ثلاث قيم للاختيار • True • False • Null	<code>IsThreeState="True"</code>	<code>IsThreeState="Value"</code>
<input checked="" type="checkbox"/> check box	<code>HorizontalAlignment="Center"</code>	<code>HorizontalAlignment="Value"</code>
<input checked="" type="checkbox"/> check box	<code>BorderBrush="Blue"</code>	<code>BorderBrush="Color"</code>

الجدول التالي يحوي اهم خصائص مربعات الاختيار.

الاحداث الرئيسية لمربع النص:

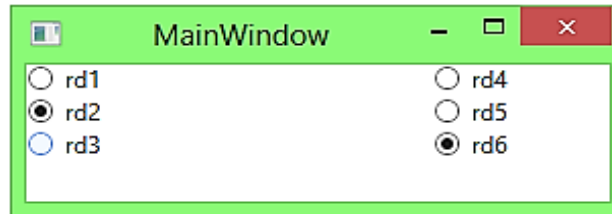
الحدث	الشرح
Checked	يحدث عندما يختار المربع
Unchecked	يحدث عندما لا يختار المربع
Indeterminate	يحدث عند اختيار الحالة الثالثة فقط وهي حالة عدم تعيين
Click	يحدث عن الضغط على مربع الاختيار ايأ كانت حالته

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

التعامل مع RadioButton:

مربعات الاختيار (واحد فقط من متعدد) هي عناصر اختيار واحد فقط من مجموعة اختيارات وهنا يجب ان نميز كل مجموعة على حدا مثال كما في الشكل التالي:



هنا المجموعة الاولى اليسارية تحوي ثلاث ازرار كلها تابعة للمجموعة ١ ، فإذا اردنا الاختيار منها يمكن اختيار مربع واحد.
اما المجموعة الثانية اليمينية فهي تابعة للمجموعة ٢ ، نلاحظ ان المجموعتين مستقلات فيما بينهما، الكود التالي يوضح الواجهة السابقة:

```
<Grid>
  <RadioButton Content="rd1" GroupName="1"></RadioButton>
  <RadioButton Content="rd2" GroupName="1" Margin="0,17,0,-17"></RadioButton>
  <RadioButton Content="rd3" GroupName="1" Margin="0,34,0,-34"></RadioButton>

  <RadioButton Content="rd4" GroupName="2" Margin="200,0,0,-17"></RadioButton>
  <RadioButton Content="rd5" GroupName="2" Margin="200,17,0,-17"></RadioButton>
  <RadioButton Content="rd6" GroupName="2" Margin="200,34,0,-34"></RadioButton>
</Grid>
```

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

التعامل مع TextBlock:

هو مربع لإظهار النص، ويتميز كل ما يتعلق بالنص بالخصائص الأساسية التالية:

الخاصية	الشرح
FontFamily	تحدد صنف الخط
FontSize	تحدد حجم الخط
FontStyle	تحدد نمط الخط (مائل – محدد بخط سفلي...)
FontWeight	تحدد ثقل الخط
Enabled	تحدد هل العنصر مفعل
ReadOnly	تحدد هل العنصر قابل للتغيير ام للقراءة فقط وهي بحالة TextBlock دوما True أي لا يمكن تغيير محتواه
TextWrapping	في حال كانت Wrap ومنه النص سوف يقفز اسطر تلقائيا ليأخذ حجم مربع النص الذي يحويه، ولن يتجاوز ذلك الحجم

من الخصائص الجديدة في WPF هي:

الخاصية	الشرح
<LineBreak/>	تقفز بالنص ضمن مربع النص والذي يكون بعدها مباشرة الى السطر التالي وتكافئ في لغات البرمجة \n
</Run>النص <Run >	تحدد ان النص الذي يقع ضمنها له خصائص نص ولون مستقل

مثال:

Hi Im Feisal Aswad

Software Engineering | **Of StackOverFlow**

Arabi Team

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



```
<Grid>
<TextBlock Name="DemoTxt" FontSize="25" TextWrapping="Wrap" FontStyle="Oblique"
FontWeight="ExtraBold" >Hi Im Feisal Aswad <LineBreak/>

    <Run FontWeight="Thin"> Software Engineering |</Run>
    <Run FontWeight="ExtraBold" Foreground="Red" > Of StackOverFlow Arabi
Team</Run>
</TextBlock>
</Grid>
```

هنا انشأنا مربع اظهر نص يحوي العبارة التالية : Hi Im Feisal Aswad Software Engineering |
of StackOverFlow Arabi Team

نلاحظ ان العبارة | Software Engineering له نمط خاص بها لأنها حددت ضمن <Run> وكذلك
العبارة التي تليها.

التعامل مع TextBox:

هو مربع لإدخال النص، ويتميز بكل ما يتعلق بالنص كما العنصر السابق.

جدول الاحداث الرئيسية المتعلقة بمربع ادخال نص:

الحدث	الشرح
KeyDown	يحدث عندما <u>نضغط</u> زر ضمن مربع النص
KeyUp	يحدث عندما <u>نرفع ضغطتنا</u> عن زر ضمن مربع النص
TextChanged	يحدث عندما يتغير النص في مربع النص

AcceptsReturn="True"	تسمح بمعرفة الاحداث الواقعة على هذا النص عند كل تغيير
MaxLength="Number"	تحدد الطول الاعظمي للنص المسموح ادخاله

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



التعامل مع PasswordBox:

هو مربع لإدخال كلمات المرور له الخاصية PasswordChar والتي تحدد حرف ليكون ظاهرا كنص بدل كلمة المرور ، كما يلي:

<pre><PasswordBox PasswordChar="#" ></PasswordBox></pre>	<p>أي ستظهر المحارف # ضمن النص مهما كانت كلمة المرور ، وذلك كي لا تكون مرئية على الشاشة</p>
--	---

التعامل مع Calendar:



الخاصية	الشرح
DisplayDayStart	تحدد تاريخ بداية الروزنامة ولا يمكن الحصول على تاريخ قبل هذا التاريخ
DisplayDayEnd	تحدد تاريخ نهاية الروزنامة ولا يمكن الحصول على تاريخ بعد هذا التاريخ
DisplayMode	تحدد نمط الاظهار عقد - سنة - شهر
SelectionMode	تحدد طريقة اختيار الأيام اما يوم واحد أو اختيار متعدد
FirsrDayOfWeek	تحدد ليوم الأول في الاسبوع
isTodayHighlighted	تحدد ان اليوم سوف يظهر بلون فاتح مميز على الروزنامة

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

التعامل مع DatePicker:



الخاصية	الشرح
SelectedDateFormat	تحدد طريقة اظهار التاريخ ضمنها اما مفصل او مختصر
IsDropDownOpen	تبقى القائمة منسدلة لإظهار واختيار تاريخ

التعامل مع Slider:



الخاصية	الشرح
IsDirectionReversed	يعكس اتجاه البداية والنهاية للزلافة
SmallChange	يحدد قيمة التغيرات الصغيرة
largeChange	يحدد قيمة التغيرات الكبيرة
Orientation	يحدد اتجاه الزلافة افقي -عامودي

التعامل مع ProgressBar:



الخاصية	الشرح
Value	تحدد قيمة الاكتمال من ١٠٠٪
IsIndeterminate	تحدد ان يبقى مستطيل الانتظار بحالة دوران

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

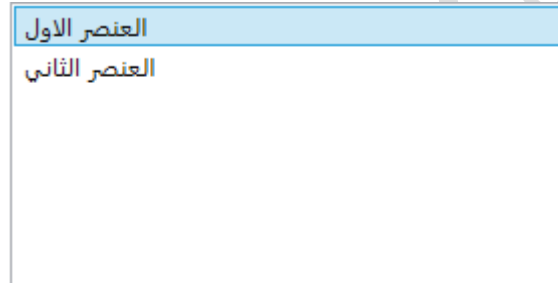
التعامل مع MediaElement:

يستخدم هذا العنصر لعرض مقطع فيديو.

الخاصية	الشرح
Volume	تحدد قيمة ارتفاع الصوت للمقطع
IsMuted	تحدد هل الصوت مسموع ام كتم

التعامل مع ListBox:

بالبداية نذكر ان الـ ListBox هو عبار عن مربع تظهر فيه العديد من الخيارات ، يمكن إضافة خيارات أخرى اثناء التصميم او التنفيذ..



كل عنصر من عناصر الـ ListBox يسمى ListBoxItem.

الجدول التالي يوضح الفرق بين إضافة عنصر لمربع القائمة او الـ ListBox عن طريق Xaml و الكود البرمجي.

<pre><ListBox Margin="72,0,157,0"> <ListBoxItem>العنصر الاول</ListBoxItem> <ListBoxItem>العنصر الثاني</ListBoxItem> </ListBox></pre>
<pre>myList.Items.Add("العنصر الاول"); myList.Items.Add("العنصر الثاني");</pre>

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

الخصائص الأساسية للـ ListBox:

الخاصية	الشرح
SelectedIndex	يعيد رقم السطر المحدد حيث يبدأ العد من الصفر او يعيد -١ في حال لم يحدد شيء.
SelectedItem	يعيد كائن يمثل العنصر الأول المحدد او null في حال عدم التحديد.
SelectedValue	يعيد قيمة العنصر الأول المحدد او null في حال عدم التحديد.
SelectionMode	له ثلاث أنماط تحديد Single يحدد عنصر فقط و multiple يحدد اكثر من عنصر و Extended يحدد عدة عناصر بشكل متتالي.

تنويه

لتحديد اكثر من عنصر بشكل متتالي عليك الضغط على اول عنصر مع زر Shift ثم اخر عنصر تريده ومنه تختار كل العنصر التي بينها بما فيها.

مثال:

في مربع النص التالي نجد كل عنصر عبارة عن صورة وبجانبها شرح كيف يمكن تحقيق ذلك.



تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

الحل ببساطة هي استخدام عنصر حاوي لكل عنصر من عناصر القائمة بحيث ان العنصر الحاوي StackPanel يحوي داخله الصورة والنص المجاوره لها، الكود التالي يوضح عملية التصميم:

```
<Grid>
  <ListBox Margin="72,0,157,0" Name="myList"
  SelectionChanged="myList_SelectionChanged">
    <StackPanel Orientation="Horizontal">
      <Image Source="E:\android.png" Name="Android" Stretch="Fill" Height="102"
      Width="101"></Image>
      <TextBlock TextWrapping="Wrap" Width="176"><Run FontWeight="ExtraBold">
      ANDROID </Run><LineBreak></LineBreak> يمكنك شروحات على للحصول واللوحة الجواله للاجهزة تشغيل نظام هو
      StackOverFlow Arabi </TextBlock>
    </StackPanel>
    <StackPanel Orientation="Horizontal">
      <Image Source="E:\go.png" Name="Go" Stretch="Fill" Height="102"
      Width="101"></Image>
      <TextBlock TextWrapping="Wrap" Width="176"><Run FontWeight="ExtraBold">
      Go </Run><LineBreak></LineBreak> يرجى للمزيد جوجل شركة من مدعومة حديثة اجرائية لغة هي
      StackOverFlow Arabi </TextBlock>
    </StackPanel>
    <StackPanel Orientation="Horizontal">
      <Image Source="E:\C++.jpg" Name="C++" Stretch="Fill" Height="102"
      Width="101"></Image>
      <TextBlock TextWrapping="Wrap" Width="176"><Run FontWeight="ExtraBold">
      C++ </Run><LineBreak></LineBreak> يمكن للمزيد C بعد الموجهة البرمجة واساس اللغات اقدم من تعتبر موجهة لغة هي
      StackOverFlow Arabi </TextBlock>
    </StackPanel>
  </ListBox>
</Grid>
```

الان نريد ان يظهر لنا اسم كل تقنية او لغة برمجية عند الضغط عليها ، الحل هنا برمجي كالتالي:

```
private void myList_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
  String name =
  ((StackPanel)myList.SelectedItem).Children.OfType<Image>().FirstOrDefault<Image>().Name.ToString();
  MessageBox.Show(name);
}
```

وهنا عند النقر على أي صورة سوف يظهر اسم التقنية التي تمثلها.

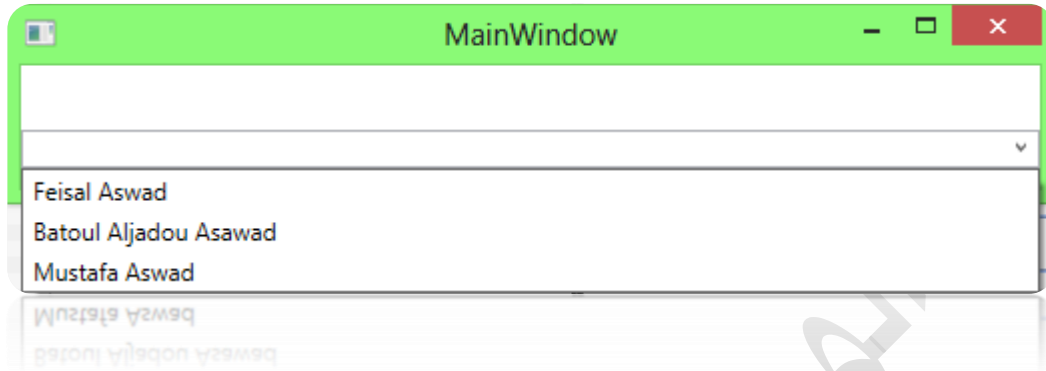
تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



التعامل مع ComboBox:

يمكن الـ ComboBox المستخدم من اختيار عنصر واحد فقط من قائمة منسدة، وله نفس الخصائص للـ ListBox المذكورة سابقاً، كما يمكن ان يحوي على صور ونصوص والخ.....



```
<Window x:Class="My_First_Project.MainWindow"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" Title="MainWindow" Height="100"
Width="525">
    <Grid>
        <ComboBox IsEditable="True" IsReadOnly="True" Margin="0,32,0,10" >
            <ComboBoxItem>Feisal Aswad</ComboBoxItem>
            <ComboBoxItem>Batoul Aljadou Asawad</ComboBoxItem>
            <ComboBoxItem>Mustafa Aswad</ComboBoxItem>
        </ComboBox>
    </Grid>
</Window>
```

للـ ComboBox خاصيتين رئيسيتين مرتبطتان هما الـ IsReadOnly و الـ IsEditable ويمكن فهمها كما يبين الجدول التالي:

IsReadOnly	IsEditable	الشرح
False	False	(الحالة الافتراضية) لا يمكن النسخ او التعديل في محتوى العنصر.
False	True	يمكن نسخ محتوى العنصر و تعديله.
True	False	لا يمكن النسخ او التعديل في محتوى العنصر.
True	True	يمكن فقط نسخ محتوى العنصر ولا يمكن تعديله.

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



التعامل مع TreeView:

يظهر هذا العنصر المعلومات على شكل هيكل شجرة وفروع ويمكن اظهار محتويات كل عقدة او اغلاقها ، كما تحوي عنصر من نوع TreeViewItem والذي يمكن ان يكون صورة او نص او....

```

  ▲ wpf
    StackPanel
    Grid
    WrapPanel
  ▲ C++
    For loop
    while loop

```

والكود التصميمي الموافق للشكل السابق هو:

```

<Window x:Class="My_First_Project.MainWindow"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" Title="MainWindow"
Height="300" Width="525">
  <Grid>
    <TreeView>
      <TreeViewItem Header="wpf" Tag="Group">
        <TreeViewItem Header="StackPanel"></TreeViewItem>
        <TreeViewItem Header="Grid"></TreeViewItem>
        <TreeViewItem Header="WrapPanel"></TreeViewItem>
      </TreeViewItem>
      <TreeViewItem Header="C++" Tag="Group">
        <TreeViewItem Header="For loop"> </TreeViewItem>
        <TreeViewItem Header="while loop"></TreeViewItem>
      </TreeViewItem>
    </TreeView>
  </Grid>
</Window>

```

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



مثال:

المطلوب تصميم TreeView برمجياً وإظهار رسالة اسم كل عنصر منها عند الضغط عليه.

```

└─ Class 1
   1
└─ Class 2
   2

```

```

TreeViewItem firstNode = new TreeViewItem();
firstNode.Header = "Class 1";
firstNode.Tag = "Group";

TreeViewItem firstNodeChild = new TreeViewItem();
firstNodeChild.Header = "1";
firstNode.Items.Add(firstNodeChild);

TreeViewItem SecoundNode = new TreeViewItem();
SecoundNode.Header = "Class 2";
SecoundNode.Tag = "Group";

TreeViewItem SecoundNodeChild = new TreeViewItem();
SecoundNodeChild.Header = "2";
SecoundNode.Items.Add(SecoundNodeChild);

MyTreeVeiw.Items.Add(firstNode);
MyTreeVeiw.Items.Add(SecoundNode);

```

والآن لإظهار نص كل عنصر عند الضغط عليه نكتب:

```

private void MyTreeVeiw_SelectedItemChanged(object sender, RoutedPropertyChangedEventArgs<object>
e)
{
string s =
((TreeViewItem)MyTreeVeiw.SelectedItem).GetValue(TreeViewItem.HeaderProperty).ToString();
MessageBox.Show(s);
}

```

تأليف فيصل الأسود | مهندس برمجيات

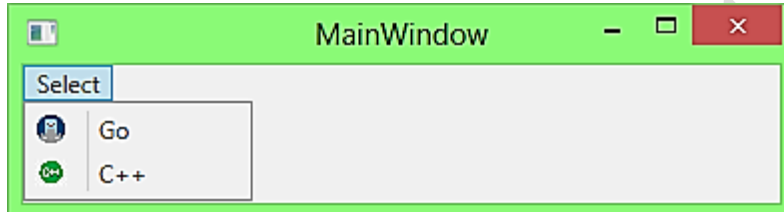
يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



التعامل مع الـMenu:

مازالت التطبيقات حتى يومنا هذا تستخدم القوائم، حيث يمكن وضعها أينما تريد في الواجهة فتوفر وصول سريع للخيارات، وغالباً ما توضع القائمة في الزاوية العليا اليسرى.

العنصر المستخدم في القوائم هو الـMenuItem وكل خيار ضمنه يسمى MenuItem يمكن وضع النص التعريفي لكل قائمة او عنصر فيها عن طريق الخاصية Header كما يمكن إضافة صورة مصغرة عن طريق الخاصية icon.



والكود المقابل:

```
<Menu IsMainMenu="True" Margin="0,0,-8,0" >
  <MenuItem Header="Select">
    <MenuItem Header="Go" Click="MenuItem_Click_1">
      <MenuItem.Icon>
        <Image Source="E:\Go.png"></Image>
      </MenuItem.Icon>
    </MenuItem>

    <MenuItem Header="C++" Click="MenuItem_Click">
      <MenuItem.Icon>
        <Image Source="E\C++.jpg"></Image>
      </MenuItem.Icon>
    </MenuItem>
  </MenuItem>
</Menu>
```

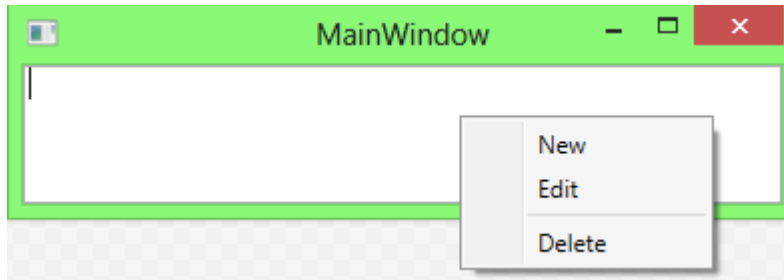
وهنا نلاحظ ان لكل عنصر قائمة حدث ينفذ عند الضغط على ذلك العنصر.

لدينا أيضاً نوع اخر من القوائم تلك التي تظهر عند الضغط بالزر الأيمن على احد العناصر ويسمى هذا النوع بالـContextMenu.

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

مثال:

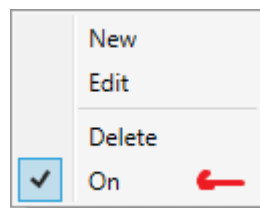


نلاحظ ظهور قائمة تحوي ثلاث عناصر لكل منها عمل ، الكود المقابل:

```
<Grid>
  <TextBox>
    <TextBox.ContextMenu>
      <ContextMenu>
        <MenuItem Header="New"></MenuItem>
        <MenuItem Header="Edit"></MenuItem>
        <Separator></Separator>
        <MenuItem Header="Delete"></MenuItem>
      </ContextMenu>
    </TextBox.ContextMenu>
  </TextBox>
</Grid>
```

نلاحظ وجود فاصل بين الخيارين الأوليين والخيار الثالث وهذا يتحقق بإضافة ما يسمى بال- Separator.

كما يمكن جعل عنصر يقبل الاختيار او عدم الاختيار عند طريق الخاصية IsCheckable وسيصبح كما في الشكل التالي:



تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

التعامل مع الـ ScrollView:

هو عنصر يساعد على قراءة المعلومات في حال كانت لا تتسع مع ابعاد النافذة.

ولديه عدة اشكال للتعامل معه:

`HorizontalScrollBarVisibility=""`

الخاصية	الشرح
Auto	يظهر عند وجود معلومات غير متسعة مع النافذة، ويختفي عند عدم ذلك (افتراضي)
Visible	دائماً ظاهر
Disabled	دائماً مخفي ولا يمكن التعامل معه حتى برمجياً
hidden	مخفي ولكن يمكن التعامل معه برمجياً

التوابع الممكنة:

LineDown()
LineUp()
LineRight()
LineLeft()
ScrollToRightEnd()
ScrollToLeftEnd()
ScrollToTop()
ScrollToBottom()

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



التعامل مع الـ TabControl:

يستخدم الـ TabControl لاختيار صفحة من عدة صفحات او تبويب معين كما في الشكل.

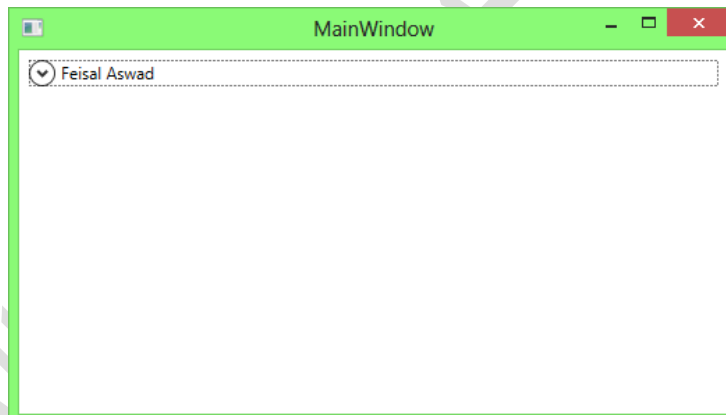


البرنامج التالي يوافق لـ TabControl السابق:

```
<TabControl>
  <TabItem Header="Subject" Name="SubjectTab"></TabItem>
  <TabItem Header="Expert Name" Name="expertTab"></TabItem>
</TabControl>
```

التعامل مع الـ Expander:

هو عبارة عن منطقة يمكن ان تتوسع وتتقلص باتجاه معين.



تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

والكود الموافق:

```
<StackPanel>
  <Expander Margin="5" Header="Feisal Aswad" >
    <StackPanel>
      <Image Source="E:\Go.Png" Stretch="None"></Image>
    </StackPanel>
  </Expander>
</StackPanel>
```

يحوي العنصر Expander عدد من الاحداث منها:

الحدث	الشرح
Collapsed	يحدث عندما يقلص العنصر
Expanded	يحدث عندما يوسع العنصر

فيصل الأسود

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



المصادر Resources

تأليف فيصل الأسود | مهندس برمجيات
يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



المصادر Resources:

تقدم لغة Xaml امكانية تعريف مصادر خاصة منها للأنماط او للكائنات الخارجية او عناصر البيانات ويمكن ان نعرف عنها في عدة مواضع

حيث تجعل تلك العناصر طريقة التصميم:

- اكثر فعالية:
ومن خلالها يمكن تعريف كائن واستخدامه في عدة اماكن ضمن الكود سواء في وقت التصميم ووقت التنفيذ
- اكثر قابلية للصيانة :
حيث تعديل عنصر واحد مستخدم عدة مرات في الكود افضل من تعديله لو كان عدة عناصر
- اكثر تلائم حيث المصادر يمكن ان تخزن بشكل مستقل عن البرنامج ويمكن تعديلها واعادة توضعها ضمن البرنامج في اي وقت.

كل عنصر من العناصر يزود بعنوان المصدر الخاص به وهو عبارة عن سلسلة تمثل المصدر المحدد و كل عنصر يظهر عدد من المصادر، وكل عنصر يمكن ان يطبق عليه مصدره و مصدر اباه و اجداده التي تحتويه.

```
<UserControl>
  <UserControl.Resources>
    <LinearGradientBrush x:Key="BackgroundBrush">
      <LinearGradientBrush.GradientStops>
        <GradientStop Offset="0" Color="White"></GradientStop>
        <GradientStop Offset="0.25" Color="Yellow"></GradientStop>
        <GradientStop Offset="0.5" Color="Orange"></GradientStop>
        <GradientStop Offset="1" Color="Blue"></GradientStop>
      </LinearGradientBrush.GradientStops>
    </LinearGradientBrush>
  </UserControl.Resources>
  <Grid x:Name="LayoutRoot" Background="White">
    <Border Background="{StaticResource BackgroundBrush}"></Border>

  </Grid>
</UserControl>
```

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



المصادر تظهر ضمن UserControl.Resources، اسم المصدر يعبر عن وظيفته ويكتب ضمن x:key للاستدعاء المصدر نكتب الكلمة StaticResource للبحث ضمن المصادر كما يلي:

Background="{StaticResource BackgroundBrush}"

المصادر الثابتة vs الديناميكية:

المصدر الثابت يحمّل فقط عند التشغيل للبرنامج، واستبداله ضمن الكود لم يغير شيء أو لن يظهر بينما استبدال المصادر الديناميكية يظهر تأثيره مباشرة في وقت التنفيذ ولكن المصادر المتغيرة تسبب ببطء لذلك طالما ان المصادر لن تتغير نستخدم مصادر ثابتة. المصادر الثابتة يجب ان يتم تعريفها قبل استخدامها في التصميم كل عنصر يستخدم مصدره الخاص في حال وجود تعارض، ودوما ينظر العنصر لمصدره ثم مصدر ابيه وهكذا.

```
<UserControl>
  <UserControl.Resources>
    <LinearGradientBrush x:Key="BackgroundBrush">
      <LinearGradientBrush.GradientStops>
        <GradientStop Offset="0" Color="White"></GradientStop>
        <GradientStop Offset="0.25" Color="Yellow"></GradientStop>
        <GradientStop Offset="0.5" Color="Orange"></GradientStop>
        <GradientStop Offset="1" Color="Blue"></GradientStop>
      </LinearGradientBrush.GradientStops>
    </LinearGradientBrush>
  </UserControl.Resources>
  <Grid x:Name="LayoutRoot" Background="White">
    <StackPanel >
      <StackPanel.Resources >
        <SolidColorBrush x:Key="StackPanelBackgroundBrush" Color="Aqua">
          </SolidColorBrush>
        </StackPanel.Resources>
        <Button Content="first" Margin="10" Background="{StaticResource StackPanelBackgroundBrush}"></Button>
        <Button Content="first" Margin="10" Background="{StaticResource BackgroundBrush}"></Button>
      </StackPanel>
    </Grid>
  </UserControl>
```

هنا نلاحظ ان الزر الأول تمكن من الوصول لمصدر ابيه وهو المصدر الخاص بـ StackPanel وهو StackPanelBackgroundBrush

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



وحسب التعريف السابق للمصدر داخل العنصر StackPanel سوف يتم استدعائه كما يلي:

```
<StackPanel.Background>
<StaticResource ResourceKey="StackPanelBackgroundBrush"></StaticResource>
</StackPanel.Background>
```

الآن لنفرض ماذا سيحدث لو تعارضت أسماء المصادر مع بعضها أو تكررت.

هنا الجواب سوف يستخدم المصدر المحلي، المثال التالي يوضح كيف تعارضت المصادر واي المصادر التي تم اخذها بعين الاعتبار.

```
<UserControl>
  <UserControl.Resources>
    <LinearGradientBrush x:Key="BackgroundBrush">
      <LinearGradientBrush.GradientStops>
        <GradientStop Offset="0" Color="White"></GradientStop>
        <GradientStop Offset="0.25" Color="Yellow"></GradientStop>
        <GradientStop Offset="0.5" Color="Orange"></GradientStop>
        <GradientStop Offset="1" Color="Blue"></GradientStop>
      </LinearGradientBrush.GradientStops>
    </LinearGradientBrush>
  </UserControl.Resources>
  <Grid x:Name="LayoutRoot" Background="White">
    <StackPanel >
      <StackPanel.Resources >
        <SolidColorBrush x:Key="StackPanelBackgroundBrush" Color="Aqua">
          </SolidColorBrush>
        <SolidColorBrush x:Key="BackgroundBrush" Color="Blue">
          </SolidColorBrush>
        </StackPanel.Resources>

        <Button Content="first" Margin="10" Background="{StaticResource
StackPanelBackgroundBrush}"></Button>
        <Button Content="first" Margin="10" Background="{StaticResource
BackgroundBrush}"></Button>
      <StackPanel.Background>
        <StaticResource
ResourceKey="StackPanelBackgroundBrush"></StaticResource>
      </StackPanel.Background>
    </StackPanel>
  </Grid>
</UserControl>
</Window>
```

هنا نلاحظ ان الرز يملك المصدر **BackgroundBrush** والموجود في المصادر الخارجية بالاعلى وفي الـ StackPanel وهنا نلاحظ انه اخذ مصدر اباه مصدر الـ StackPanel.

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



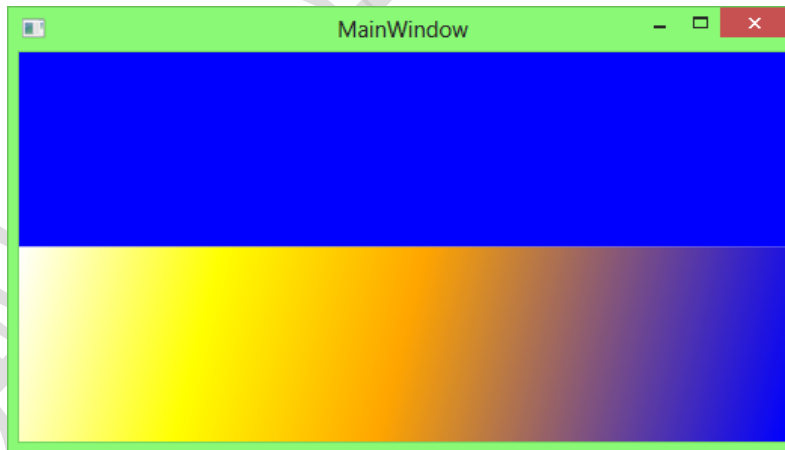
وضع المصادر في ملف البرنامج APP.Xaml File:

في المثال التالي سوف نرى ان العنصر StackPanel الأول يستخدم مصدره الخاص ، انما الـ StackPanel الثاني يستخدم مصدر بنفس اسم المصدر للأول ولكن كما يظهر في الصورة تأثير مختلف تماماً.

```
<UserControl>
  <Grid x:Name="LayoutRoot" Background="White">
    <Grid.RowDefinitions>
      <RowDefinition></RowDefinition>
      <RowDefinition></RowDefinition>
    </Grid.RowDefinitions>
    <StackPanel >
      <StackPanel.Resources >
        <SolidColorBrush x:Key="BackgroundBrush" Color="Blue"></SolidColorBrush>
      </StackPanel.Resources>

      <StackPanel.Background>
        <StaticResource ResourceKey="BackgroundBrush"></StaticResource>
      </StackPanel.Background>
    </StackPanel>
    <StackPanel Grid.Row="1" Background="{StaticResource BackgroundBrush}">

  </StackPanel>
</Grid>
</UserControl>
```



الفكرة هنا اننا استخدمنا مصدر مخزن في الملف App المتضمن في كل برنامج WPF والذي يساعد على تنظيم المصادر.

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

الترميز التالي يعبر عن محتويات الملف App File الذي يحوي مصدر العنصر الثاني.

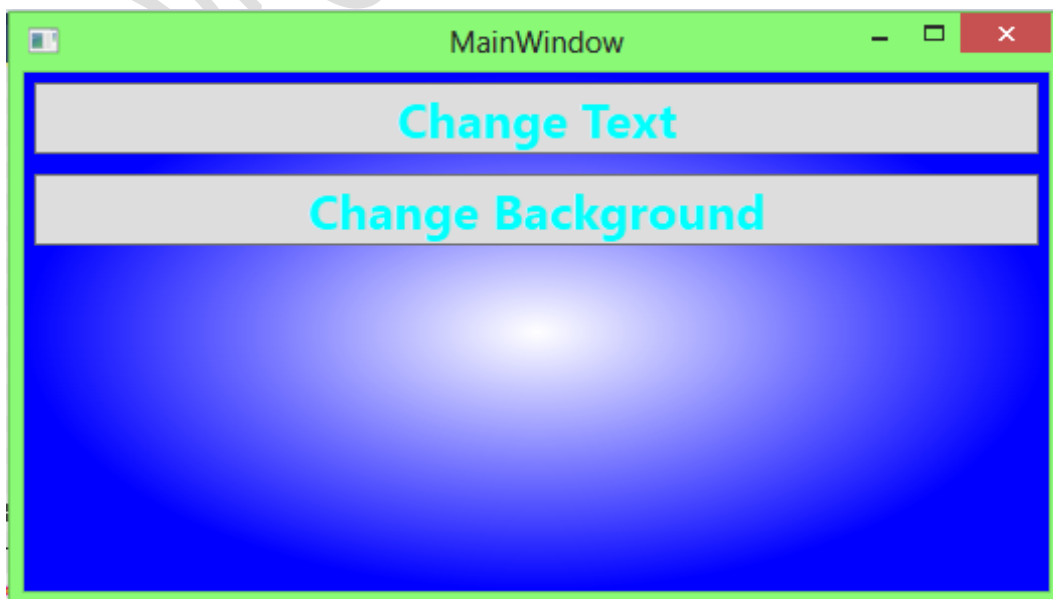
```
<Application x:Class="My_First_Project.App"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  StartupUri="MainWindow.xaml">
  <Application.Resources>
    <ResourceDictionary>
      <LinearGradientBrush x:Key="BackgroundBrush">
        <LinearGradientBrush.GradientStops>
          <GradientStop Offset="0" Color="White"></GradientStop>
          <GradientStop Offset="0.25" Color="Yellow"></GradientStop>
          <GradientStop Offset="0.5" Color="Orange"></GradientStop>
          <GradientStop Offset="1" Color="Blue"></GradientStop>
        </LinearGradientBrush.GradientStops>
      </LinearGradientBrush>
    </ResourceDictionary>
  </Application.Resources>
</Application>
```

وهكذا نرى انه يمكننا تخزين المصادر في ملف الـ APP.

التحكم بالمصادر برمجياً:

لابد ان نذكر انه يمكن التحكم بالمصادر خارجياً طالما هي مصادر ديناميكية وغير مجمدة.

المثال التالي يوضح واجهة تحوي زرین الأول يقوم بتغيير اللون الازرار والثاني يغير الخلفية وسنجد كيف تم ذلك برمجياً.



تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



والكود التالي يمثل تصميم الواجهة:

```
<UserControl>
  <UserControl.Resources>
    <SolidColorBrush x:Key="Textbrush" Color="Aqua"></SolidColorBrush>
  </UserControl.Resources>
  <Grid Background="White">
    <StackPanel Background="{DynamicResource BackgroundBrush}">
      <Button Content="Change Text" Margin="5" FontSize="24" FontWeight="Bold"
Foreground="{StaticResource Textbrush}" Click="Button_Click">
    </Button>
      <Button Content="Change Background" Margin="5" FontSize="24"
FontWeight="Bold" Foreground="{StaticResource Textbrush}" Click="Button_Click_1">
    </Button>
    </StackPanel>

  </Grid>
</UserControl>
```

نلاحظ ان الازرار تأخذ المصدر **Textbrush** المعرف ضمن الـ **StackPanel** ولكن الخلفية تأخذ المصدر **BackgroundBrush** المعرف في الملف App كما يلي:

```
<Application.Resources>
  <ResourceDictionary>
    <RadialGradientBrush x:Key="BackgroundBrush">
      <RadialGradientBrush.GradientStops>
        <GradientStop Offset="0" Color="White"></GradientStop>
        <GradientStop Offset="1" Color="Blue"></GradientStop>
      </RadialGradientBrush.GradientStops>
    </RadialGradientBrush>
  </ResourceDictionary>
</Application.Resources>
```

الكود التالي هو كود تبديل لون الازرار الى الأحمر برمجياً:

```
Button btn = (Button)sender;
var brush = (SolidColorBrush)(btn.TryFindResource("Textbrush"));
if(brush!=null)
{
  brush.Color = Colors.Red;
}
```

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



بالمثل الكود التالي هو لكود تغيير لون الخلفية:

```

Button btn=(Button)sender;
RadialGradientBrush
brush=(RadialGradientBrush)(btn.TryFindResource("BackgroundBrush"));

if(brush!=null)
{
    Color color = brush.GradientStops[1].Color;

    if(brush.IsFrozen)
    {
        RadialGradientBrush newBrush = brush.Clone();
        newBrush.GradientStops[1].Color = brush.GradientStops[0].Color;
        newBrush.GradientStops[0].Color = color;
        Application.Current.Resources["BackgroundBrush"] = newBrush;
    }
    else
    {
        brush.GradientStops[1].Color = brush.GradientStops[0].Color;
        brush.GradientStops[0].Color = color;
    }
}

```

هنا يجب التنويه اننا ناقشنا حالتين الأولى عندما يكون المصدر مجمد كما في حالة الشرط الأول ومنه نأخذ نسخة من كائن المصدر ونبدل الألوان في النسخة ثم نسند المصدر الجديد المعدل. اما في الحالة الثانية وكون المصدر غير مجمد نبدل بشكل مباشر. وأيضا يجب ان ننتبه ان المصدر المعدل هو من نوع DaynamicResource حصراً.

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



النمط Styles

المهندس فيصل الأسود
فيصل الأسود

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



الأنماط: Styles:

رأينا كيف ان المصادر جعلت من السهل تنظيم وإدارة الكائنات باستقلالية وكيف يمكن إعادة استخدامها ، كما رأينا ان المصادر يمكنها ان تحوي كائنات مختلفة واغلبها يستخدم من اجل الأنماط.

النمط هو مجموعة من الخصائص بقيم محددة يمكن تطبيقها مباشرة على أي عنصر او كائن وتفيد لعدم تكرار إعادة كتابة الخصائص اكثر من مرة حيث تكتب مرة وتطبق عدة مرات للعناصر الباقية ، وغالباً توضع بعد العناصر.

الـ XAML تزودنا بتوابع مثل التي تزودها CSS و HTML، الان لنفرض انك تريد ان تشارك في الخط والخلفية لكل زر من ازرار برنامجك وهنا الالاسهل ان تنشئ نمط وتطبيقه على كل الازرار بدلاً من وضع نفس الخصائص لكل زر على حدى.

مثال:



من المهم ان نذكر ان النمط يطبق فقط خصائص من نوع `DependencyProperty` ويوضع ضمن وسوم الـ `UserControl.Resource` ، كما يجب الالخذ بعين الالاعتبار العنصر المراد تطبيق هذا النمط عليه بالخاصية `TargetType`.

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

```

<UserControl>
  <UserControl.Resources>
    <Style x:Key="ColorfulButtonStyle" TargetType="Button">
      <Setter Property="Background" Value="Blue" /> <Setter
Property="FontWeight" Value="ExtraBold" />
      <Setter Property="FontSize" Value="25" />
      <Setter Property="Background">
        <Setter.Value>
          <LinearGradientBrush EndPoint="0,1" StartPoint="1,0">
            <GradientStop Offset="0.1" Color="Blue"/>
            <GradientStop Offset="0.5" Color="Aqua"/>
          </LinearGradientBrush>
        </Setter.Value>
      </Setter>
    </Style>
  </UserControl.Resources>
  <StackPanel>
    <Button Content="Button 1" Margin="20" Style="{StaticResource
ColorfulButtonStyle}"></Button>
    <Button Content="Button 2" Margin="20" Style="{StaticResource
ColorfulButtonStyle}"></Button>
    <Button Content="Button 3" Margin="20" Style="{StaticResource
ColorfulButtonStyle}"></Button>
  </StackPanel>
</UserControl>

```

توضع جميع الخصائص ضمن الـ <setter> وبفرض الخاصية طويلة. نضع

```

<Setter Property="Background">
  <Setter.Value>
    <LinearGradientBrush EndPoint="0,1" StartPoint="1,0">
      <GradientStop Offset="0.1" Color="Blue"/>
      <GradientStop Offset="0.5" Color="Aqua"/>
    </LinearGradientBrush>
  </Setter.Value>
</Setter>

```

وأيضاً لتطبيق النمط نكتب:

```
Style="{StaticResource ColorfulButtonStyle}"
```

إذا لم نستخدم الـ TargetType نكتب الخاصية كما يلي:

```
Class.Property="Background"
```

بالمثل إذا لم نستخدم X:key سوف يطبق النمط على كل عناصر الواجهة التي في النمط الهدف.

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

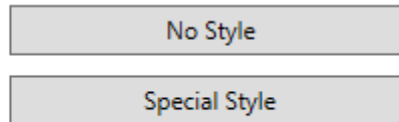


يمكن استخدام أداة Karl shfflets power toys التي تحول خصائص العناصر الى انماط

تتويبه

يمكن استبدال النمط برمجياً في وقت التنفيذ عند استخدام نمط ديناميكي:

مثال:



ليكن لدينا السابق والذي يوضح واجهة تحوي زرین الأول يجعل الازرار بدون نمط والثاني يطبق نمط معين عند الضغط عليهما.

```
<Grid>
  <Border>
    <StackPanel>
      <Button Name="Btn1" Height="25" Width="200" Margin="5"
Click="Button_Click_3">No Style</Button>
      <Button Name="Btn3" Height="25" Width="200" Margin="5"
Click="Btn3_Click">Special Style</Button>
    </StackPanel>
  </Border>
</Grid>
```

الترميز السابق يطابق تصميم الواجهة .

ولدينا ايضاً الملف الخاص بالنمط Special Style بملف مستقل كما يلي:

```
<ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

  <Style x:Key="ButtonStyle" TargetType="Button">
    <Setter Property="Background" Value="Aqua"></Setter>
    <Setter Property="FontWeight" Value="Light"></Setter>
  </Style>

</ResourceDictionary>
```

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



الآن عند الضغط على الزر الأول ينفذ الكود التالي وتبقى الواجهة كما هي:

```
Style ButtonStyle = null;
ResourceDictionary rd = null;

Btn1.Style = ButtonStyle;
Btn3.Style = ButtonStyle;
```

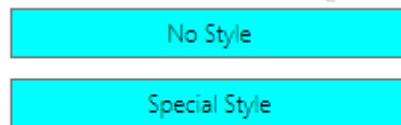
أما عند الضغط على الزر الثاني ينفذ الكود التالي وتصبح الواجهة كما مبين في الشكل في الأسفل:

```
filename= "SpecialStyle.xaml";
rd = new ResourceDictionary();
rd.Source = new Uri(fileName,UriKind.Relative);

ButtonStyle = (Style)(rd["ButtonStyle"]);

Btn1.Style = ButtonStyle;

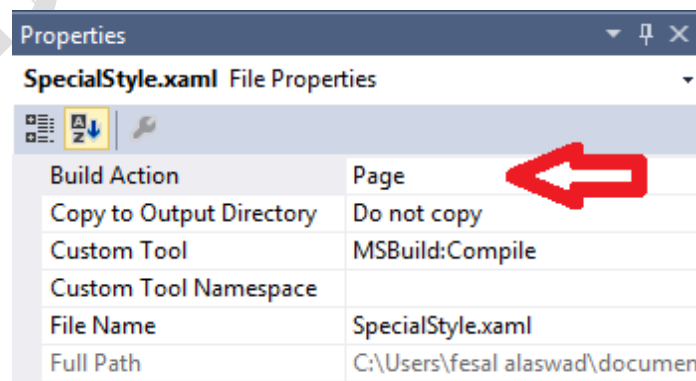
Btn3.Style = ButtonStyle;
```



نلاحظ من المثال ان استخدام النمط null يعود الى الحالة الافتراضية للعناصر.

كذلك يجب ان نذكر ان في صفحة المصدر او مصدر النمط يجب وضع حالة الخاصية

Build Action=Page



تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

يمكن للأنماط ان تستخدم المصادر كما يلي:

```
<Thickness x:Key="Borderthickness" >1,1,1,1</Thickness>
<Style x:Key="ButtonStyle" TargetType="Button">
  <Setter Property="Background" Value="Aqua"></Setter>
  <Setter Property="FontWeight" Value="Light"></Setter>
  <Setter Property="BorderThickness" Value="{StaticResource Borderthickness}"></Setter>
</Style>
```

لنفرض انه لديك مجموعة من الأنماط تملك مجموعة من الخصائص وتريد ان تنشئ نمط ثاني لاستخدام تلك الخصائص، هنا يمكن نسخها ولصقها، ولكن الأفضل في Xaml وراثتها. الوراثة في WPF تسمح لك باستخدام والاضافة على الخصائص، لدينا المثال التالي:

```
<UserControl.Resources>
<Style x:Key="FatherStyle" TargetType="Button">
  <Setter Property="Background" Value="Aqua"></Setter>
</Style>
<Style x:Key="SonStyle" TargetType="Button" BasedOn="{StaticResource FatherStyle}">
  <Setter Property="Foreground" Value="Cornsilk"></Setter>
</Style>
</UserControl.Resources>
```

هنا لدينا نمطين الأول اب للثاني لان لو طبقنا النمط الأول على زر سوف يحصل على خاصية لون الخلفية، في حين لو طبقنا النمط الثاني سوف يحصل على الخاصيتين لون النص والخلفية لانه النمط الثاني يأخذ خصائصه وخصائص اباه. تتم الوراثة في Xaml باستخدام الخاصية **BasedOn** لكن دائماً ينصح بعدم استخدام الوراثة الا في حالات الضرورة.

مماذا لو طبقنا نمط على عنصر لا يملك احد خصائص هذا النمط وهنا يتم تجاهل تلك الخصائص على العنصر.

تنويه

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



لكي نطبق نمط على كل عناصر نوع عناصر تحكم معين (ازرار مثلاً)

هنا نكتب `X.Type="Button"`

وهنا يطبق هذا النمط على كل ازرار الواجهة.

ولاستثناء احدها نكتب ضمن خصائصه

`Style="{x:null}"`

تنويه

`Style="{x:null}"`

المهندس فيصل الأسود

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



التصفح Navigation

المؤلف فيصل الأسود

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



التصفح Navigation:

تطبيقات ويندوز تعتمد على مبدأ التصفح حيث قوائم الاختيار والازرار تفتح نوافذ جديدة مستخدمة التصفح باستخدام النوافذ ، اما صفحات الويب تستخدم التصفح باستخدام الصفحات بحيث صفحة واحدة ظاهرة في نفس الوقت وتحمل كل مرة عند طلب الرابط ، يجب ان نذكر هنا ان Wpf تزود كلتا طريقتي التصفح بالصفحات والنوافذ.

صنف الـPage:

هذا الصنف مشابه ل صنف Windows ويعتبر من العناصر الحاوية ، ولكن الاختلاف الأساسي بين النوافذ والصفحات ان الصفحات تصمم للتصفح من قبل نافذة او Frame او صفحة اخرى او متصفح ويب.

صنف الـNavigationWindow

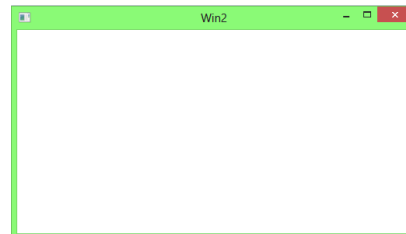
يمثل نافذة مع تصميم يجعلها قابلة للتصفح بحيث تتضمن القدرة على اظهار محتويات الويب والتصفح.

الان نذكر ان الخاصية NavigateUri هي المسؤولة عن الانتقالات بين الصفحات وتكون مرتبطة برابط معين HyperLink.

مثال:



Win1



Win2

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

عند الضغط على الزر في النافذة الأولى سوف تفتح النافذة الثانية كما يلي:

```
var Win2 = new Win2();
Win2.Shows();
```

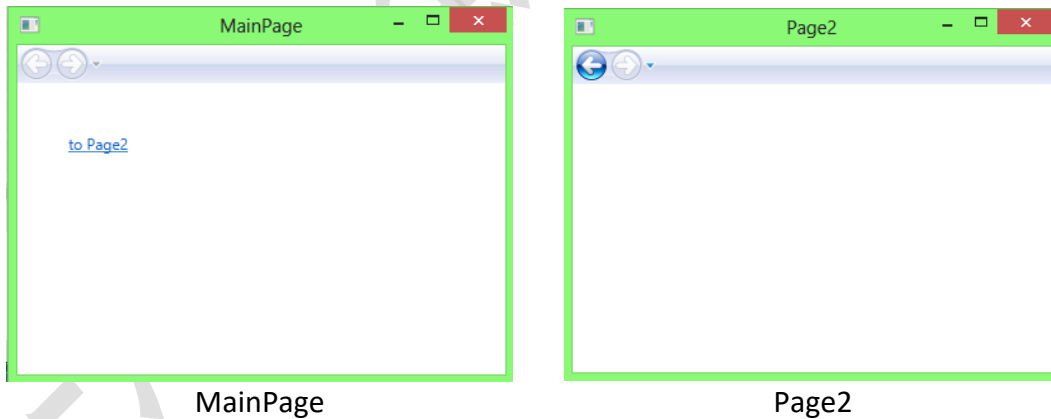
الآن نريد ان ننشئ تصفح باستخدام الصفحات وليس النوافذ:

- ننشئ مشروع WPF جديد.
- نحذف الـ MainWindow.Xaml.
- نضيف من القائمة Add عنصر جديد وهو (Wpf) Page.

مثال:

الآن سوف ننشئ صفحة رئيسية MainPage وصفحة أخرى Page2.

وهنا يجب ان نذكر ان يجب ان نغير الخاصية StartupUri في الملف App.Xaml ونضع القيمة لها StartupUri="MainPage" كي يفتح المشروع من الصفحة الأولى.



هنا عند الضغط على الرابط سوف ينتقل الى الصفحة الثانية دون ان يفتح نافذة جديدة انما ضمن نفس النافذة.

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

والكود المقابل للمثال السابق كما يلي:

```
<Page x:Class="My_First_Project.Win1"
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"
mc:Ignorable="d"
d:DesignHeight="300" d:DesignWidth="300"
WindowTitle="MainPage" WindowHeight="300" WindowWidth="400">

<TextBlock Margin="40">
<Hyperlink NavigateUri="Page2.xaml" > to Page2</Hyperlink>
</TextBlock>

</Page>
```

الـ NavigateUri يمكن ان ينتقل الى احد العناصر التالية:

- صفحة في برنامج.
- صفحة من مشروع اخر.
- صفحة ويب عادية.
- عنصر تحكم في نفس الصفحة او صفحة مختلفة (ويسمى تصفح القطع Fragmentation)

الآن ليكن لدينا المثال الذي يظهر الرابط عند الضغط عليه نتيجة تنفيذ الحدث Click="ShowUri" نفس المثال السابق تماماً ولكن بإضافة حدث للعنصر HyperLink والكود الخاص بالحدث هو:

```
String Link = ((Hyperlink)sender).NavigateUri.ToString();
MessageBox.Show(Link);
```

الفكرة هنا اننا نلاحظ انه ينفذ الحدث اولاً وهو اظهار رسالة ومن ثم ينتقل للصفحة الثانية.

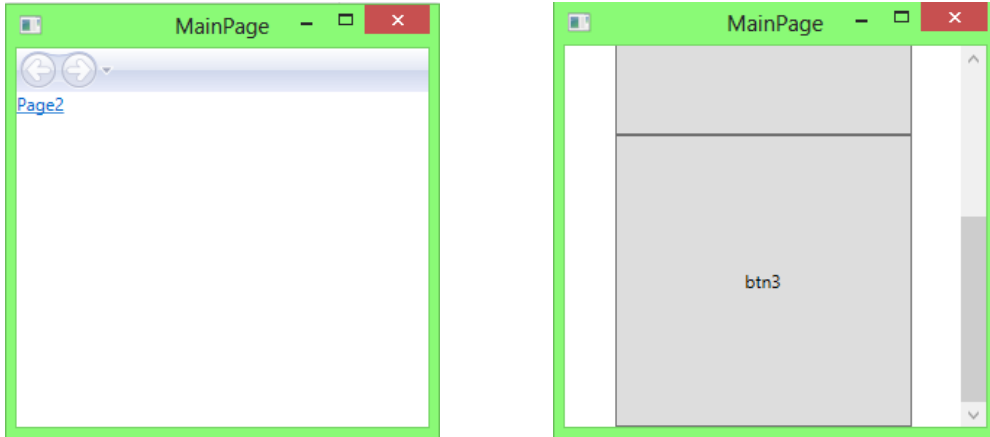
تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



تصفح القطع FragementNavigation:

عندما نضع في نهاية الرابط #ElementName ومنه سوف يذهب مباشرة الى الصفحة المختارة وبالتحديد الى العنصر المحدد وهنا نفهم اننا ليس فقط يمكننا الذهاب لصفحة انما يمكن الذهاب الى عنصر خاص فيها.



NavigateUri="Page2.xaml#btn3"

نلاحظ هنا انه انتقل من الصفحة الرئيسية الى الصفحة الثانية وبالتحديد الى موقع الزر ٣ في الصفحة.

التصفح من مكتبة خارجية:

هنا فقط نكتب الرابط بالشكل التالي:

NavigateUri="/NameLibrary; component /NamePage"

NameLibrary تمثل اسم المكتبة

NamePage تمثل اسم الصفحة

هنا لا يجب ان ننسى إضافة المكتبة كمرجع الى المشروع الرئيسي وكذلك تنفيذ Build للمكتبة.

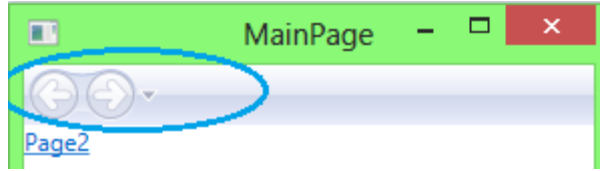
تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

الصف NavigationService:

يزود هذا الصف دعم للمتصفح متضمناً طرق تساعد التصفح.

كما رأينا ان لكل صفحة يوجد أدوات تساعد على التصفح للانتقال بين الصفحات ذهاباً واياباً بالإضافة الى احتوائها مسارات التصفح ككل.



يوجد خاصية للصفحة تمنعها من ظهور تلك الأدوات وهي `ShowsNavigationUI="False"`

وهنا لا يمكننا الرجوع للصفحة السابقة او الذهاب للصفحة التالية لان الأدوات اختفت لذلك نستخدم الصف NavigationService الذي يحل الأمور برمجياً باحتوائه على طرق وقيم واحداث تساعدني على القيام بالانتقالات.

الجدول التالية تمثل اهم ما يملكه هذا الصف:

القيمة	الشرح
CanGoBack	تمثل قيمة منطقية تشير الى إمكانية الرجوع للصفحة السابقة
CanGoForward	تمثل قيمة منطقية تشير الى إمكانية الذهاب للصفحة التالية

التابع	الشرح
GoBack	يقوم بالانتقال للخلف في حال هناك إمكانية لذلك
GoForward	يقوم بالانتقال للامام في حال هناك إمكانية لذلك

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

الحدث	الشرح
Navigation	ينفذ عند طلب التصفح
Navigated	ينفذ بعد طلب التصفح
NavigationFailed	ينفذ عند فشل التصفح

المهندس فيصل الأسود

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



الربط Binding

المهندس فيصل الأسود
فيصل الأسود

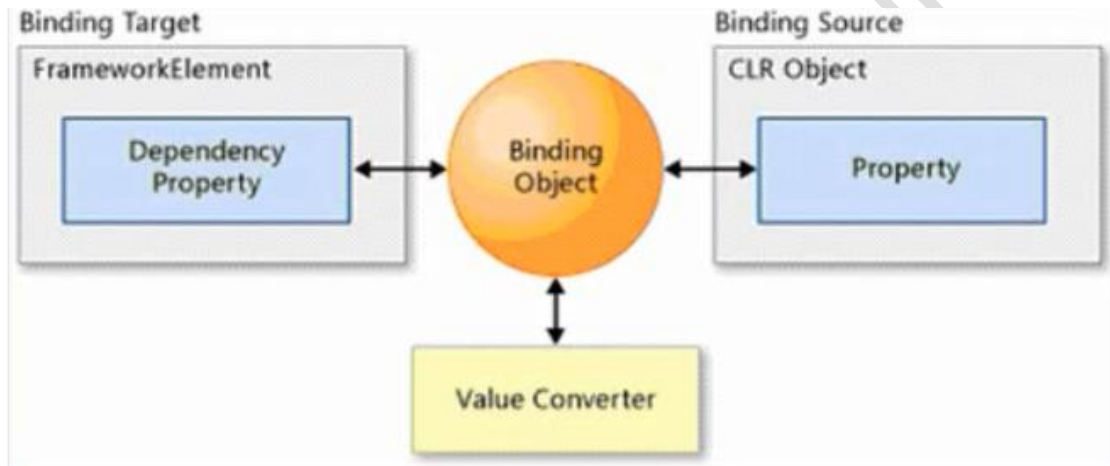
تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



الربط Binding:

أحيانا تريد تحديث معلومات عنصر من عنصر اخر او ان تظهر المعلومات تجمع كائنات ضمن قائمة والحاجة للعمل مع بيانات قاعدة البيانات Database والحل هنا باستخدام الربط Bind لربط البيانات من مصدر الى الهدف.



الصورة السابقة توضح مبدأ الربط والذي يعتمد على وضع خاصية من عنصر مصدر في خاصية ظاهرية في عنصر اخر ، وهنا نحتاج لعملية تحويل القيمة من المصدر لتناسب اظهارها على الهدف مثلاً نحتاج لمحول يحول بيانات قواعد المعطيات لتناسب للظهور على الشاشة ضمن جدول وهكذا.

مثال:

نريد ربط مربع ادخال نص ليأخذ قيمته من قيمة مزلاق Slider كما في الشكل:



تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

والكود المقابل لتحقيق عملية الربط:

```
<Slider Name="mySlider"></Slider>
<TextBox Text="{Binding ElementName=mySlider ,Path=Value, Mode=TwoWay}"
Margin="0,31,0,-31"></TextBox>
```

يمكن إضافة الخاصية StringFormat=0.00 الى الـ Binding والتي تجعل النص على شكل فاصلة مئوية.

عملية الربط تحققت بالخاصية

```
Text="{Binding ElementName=mySlider ,Path=Value, Mode=TwoWay}"
```

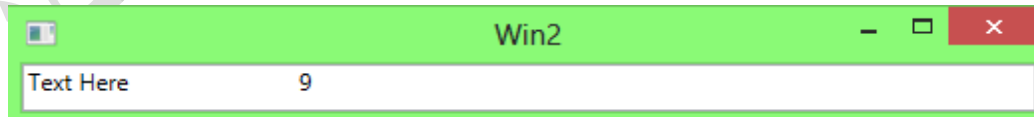
وتعني ربط النص لمربع النص TextBox بالعنصر mySlider وبالتحديد قيمته.

الجدول التالي يوضح أنماط الربط:

النمط	الشرح
One Way	تغيير قيمة المصدر تغيير قيمة الهدف فقط
Two Way	تغيير قيمة المصدر تغيير الهدف والعكس
One Way to source	تغيير قيمة الهدف تغيير قيمة المصدر فقط
One Time	تغيير قيمة المصدر تغيير قيمة الهدف فقط ومرة واحدة اثناء تحميل البرنامج

مثال:

لدينا المثال التالي الذي يربط قيمة نص بعدد احرف نص اخر كما يلي:



هنا نكتب الكود التالي:

```
<TextBox Name="MyTextBox">Text Here</TextBox>
<TextBlock Text="{Binding ElementName=MyTextBox , Path=Text.Length}"
Margin="139,1,-139,-1"></TextBlock>
```

تأليف فيصل الأسود | مهندس برمجيات

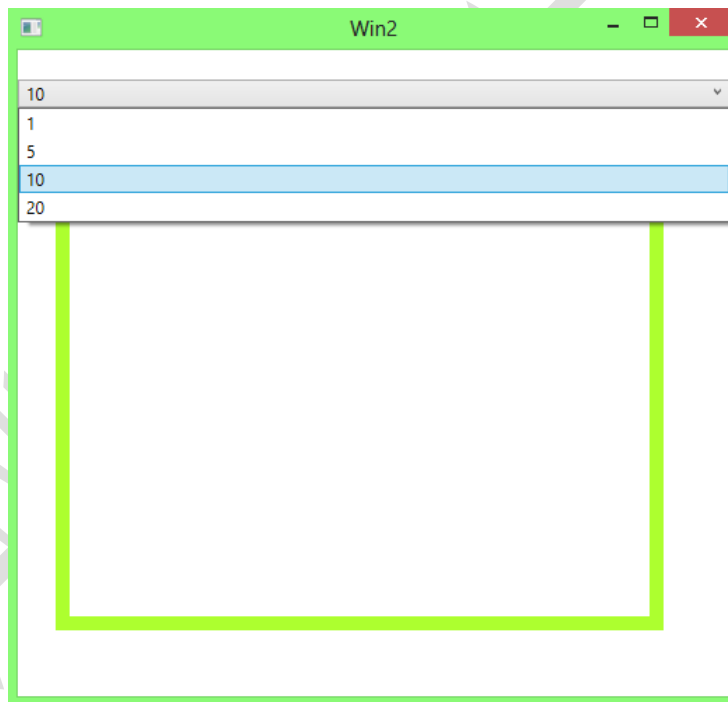
يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

او يمكن ان نكتب بالشكل التالي:

```
<TextBox Name="MyTextBox">Text Here</TextBox>
<TextBlock Margin="139,1,-139,-1">
  <TextBlock.Text>
    <Binding ElementName="MyTextBox" Path="Text.Length"></Binding>
  </TextBlock.Text>
</TextBlock>
```

الان ماذا لو احتجنا ربط المصدر مع الهدف ولكن هناك تعارض في أنواع البيانات بين الاثنين هنا نحتاج لمحاولات ValueConverter .

مثال ليكن لدينا الواجهة التي تحوي قائمة منسدلة فيها قيم و Border المطلوب هو ربط قيم القائمة مع ثخانة الـ Border كما في الشكل التالي:



تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

الحل كما في الكود التالي:

```
<Window x:Class="My_First_Project.MainWindow"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:local="clr-namespace:My_First_Project"
Title="Win2" Height="500" Width="525" Loaded="Window_Loaded" >

<UserControl>
<UserControl.Resources>
<local:ConverterClass x:Key="MyConverter"/>
</UserControl.Resources>

<Grid>
<ComboBox Height="20" Margin="0,-400,0,0" Name="MyComboBox">
<ComboBoxItem>1</ComboBoxItem>
<ComboBoxItem>5</ComboBoxItem>
<ComboBoxItem>10</ComboBoxItem>
<ComboBoxItem>20</ComboBoxItem>
</ComboBox>
<Border BorderBrush="GreenYellow"
BorderThickness="{Binding ElementName=MyComboBox,
Path=SelectedValue ,Converter={StaticResource MyConverter}}"
Margin="27,59,48,47">

</Border>
</Grid>
</UserControl>
</Window>
```

هنا عرفنا بالبداية قائمة منسدة تحوي القيم ١ - ٥ - ١٠ - ٢٠ والتي ستمثل ثخانة الـ Border فيما بعد وكذلك عرفنا Border وربطناها مع القائمة المنسدة وبالتحديد مع الخيار المحدد الحالي وبسبب اختلاف الأنواع اضطررنا الى استخدام محول قيمة MyConverter.

نعرف محول القيمة في القسم Resources ولكن يجب ان لا ننسى ان نعرف مجال البرنامج ككل في خصائص النافذة Window كما يلي :

```
xmlns:local="clr-namespace:My_First_Project"
```

وذلك لإمكانية الوصول لصنف التحويل ConverterClass.

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



```

namespace My_First_Project
{
    class ConverterClass:IValueConverter
    {
        int returnVal;
        public object Convert(object value, Type targetType, object parameter,
System.Globalization.CultureInfo culture)
        {
            if(value!=null)
            {
                String ThicknessVal = ((ComboBoxItem)value).Content.ToString();
                returnVal = Int32.Parse(ThicknessVal);
            }
            return returnVal;
        }

        public object ConvertBack(object value, Type targetType, object parameter,
System.Globalization.CultureInfo culture)
        {
        }
    }
}

```

يجب ان نعلم ان صنف التحويل دائماً يطبق الواجهة `IValueConverter` والتي تحوي
طريقتين هما: `Convert` و `ConvertBack`

و دائماً الـ `Convert` تأخذ دخل كائن محدد (حسب القائمة او العنصر المحدد) وخرجها خاصية ما
والدالة `ConvertBack` بالعكس.

هنا في حالتنا العنصر المحدد من نوع `ComboBoxItem` اخذنا منه قيمته النصية وحولناها لقيمة
ثخانة معينة ، ولا يمكن تطبيق التحويل العكسي لذلك بقيت الطريقة `ConvertBack` فارغة.

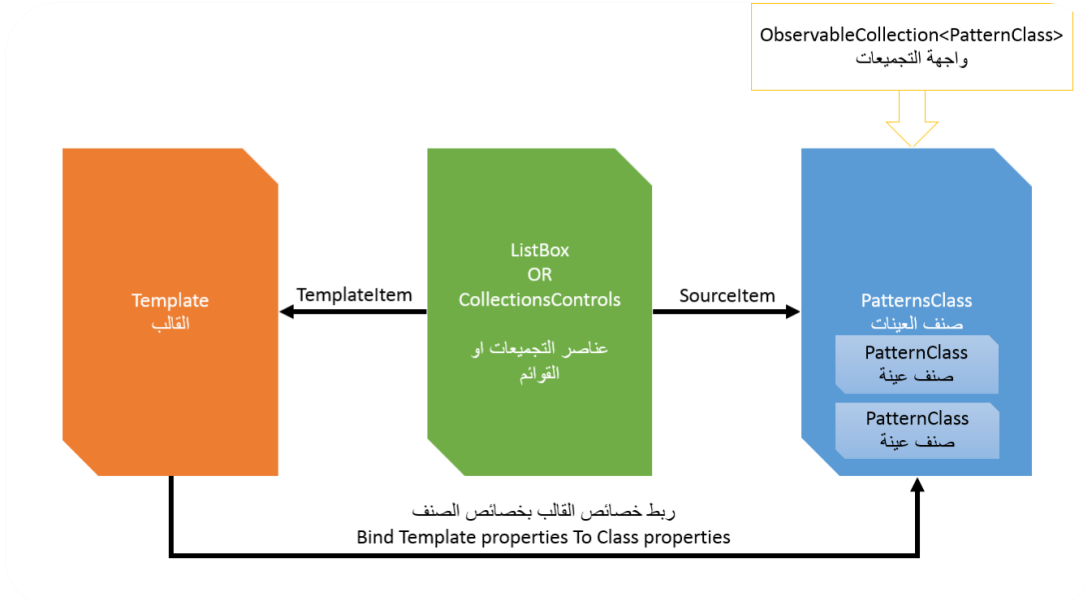
تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



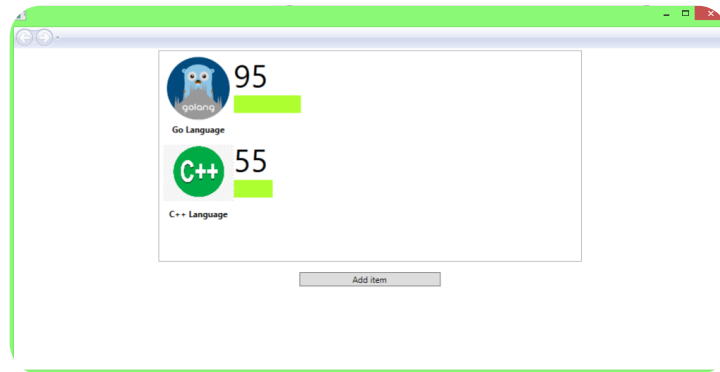
الربط للتجميعات:

ماذا لو اردنا ان نربط مجموعة من البيانات الى قائمة منسدلة او قائمة حتى هذه العناصر والتي تدعى تجميعات لها طريقة ربط مع البيانات مميزة مما يتيح لنا التمكن بقوة بتصميم القوائم بشكل احترافي.



مثال:

ليكن لدينا المثال التالي الذي يظهر قائمة تحوي لغات البرمجة على شكل (صورة - نص - تقييم - شكل بياني يمثل التقييم)



تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

يجب علينا ربط التجميعات التقييد قدر الإمكان بالمخطط السابق ضمن الخطوات التالية:

١. انشاء صنف يمثل عينة البيانات المطلوب اظهارها.

```
public class Pattern
{
    public String ImageLanguagePath { get; set; }
    public String NameLanguage { get; set; }
    public int RateLanguage { get; set; }
    public Pattern(String ImageLanguagePath, String NameLanguage, int
RateLanguage)
    {
        this.ImageLanguagePath = ImageLanguagePath;
        this.NameLanguage = NameLanguage;
        this.RateLanguage = RateLanguage;
    }
}
```

٢. انشاء صنف العينات واطافة اليه العينات كما في المثال.

```
public class Patterns : ObservableCollection<Pattern>
{
    public Patterns()
    {
        Pattern p1=new Pattern(@"E:\Go.png", "Go Language",95);
        Pattern p2=new Pattern(@"E:\C++.jpg", "C++ Language",55);
        this.Add(p1);
        this.Add(p2);
    }
}
```

يجب ان تطبق صنف التعيينات السابق الصنف `ObservableCollection` وبنفس صنف العينة كي يكون قابل للاظهار ضمن قائمة.

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



٣. إنشاء قالب ضمن قسم المصادر للبرمجية.

```

<UserControl.Resources>
  <DataTemplate x:Key="MyTemplate">

    <StackPanel Orientation="Horizontal">

      <StackPanel Orientation="Vertical">
        <Image Source="{Binding Path=ImageLanguagePath}"
Height="100" Width="100"></Image>
        <TextBlock Text="{Binding Path=NameLanguage}"
FontWeight="ExtraBold" HorizontalAlignment="Center"></TextBlock>
      </StackPanel>

      <StackPanel Orientation="Vertical">
        <TextBlock Text="{Binding Path=RateLanguage}"
FontSize="45"></TextBlock>
        <Rectangle Fill="GreenYellow" Width="{Binding
Path=RateLanguage}" Height="25"></Rectangle>
      </StackPanel>

    </StackPanel>

  </DataTemplate>
</UserControl.Resources>

```

نلاحظ ضمن القالب اننا ربطنا عناصر القالب مع خصائص الكلاس الخاص بالعينة.

فيصل الأسود

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



٤. والآن يكون التصميم النهائي.

```
<UserControl x:Class="My_First_Project.DataTemplate1"
    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"
    mc:Ignorable="d" xmlns:local="clr-namespace:My_First_Project"
    d:DesignHeight="300" d:DesignWidth="300" Name="UCDataTemplate1"
    Loaded="UCDataTemplate1_Loaded">

    <UserControl.Resources>
        <DataTemplate x:Key="MyTemplate">

            <StackPanel Orientation="Horizontal">

                <StackPanel Orientation="Vertical">
                    <Image Source="{Binding Path=ImageLanguagePath}" Height="100"
Width="100"></Image>
                    <TextBlock Text="{Binding Path=NameLanguage}"
FontWeight="ExtraBold" HorizontalAlignment="Center"></TextBlock>
                </StackPanel>

                <StackPanel Orientation="Vertical">
                    <TextBlock Text="{Binding Path=RateLanguage}"
FontSize="45"></TextBlock>
                    <Rectangle Fill="GreenYellow" Width="{Binding Path=RateLanguage}"
Height="25"></Rectangle>
                </StackPanel>

            </StackPanel>

        </DataTemplate>
    </UserControl.Resources>
    <Grid>

        <ListBox Height="300"
            Width="600"
            Margin="0,-150,0,0"
            ItemsSource="{Binding ElementName=UCDataTemplate1, Path=patterns}"
            ItemTemplate="{StaticResource MyTemplate}">

        </ListBox>

        <Button Height="20" Width="200" Click="Button_Click_1" Margin="0,200,0,0">
Add item</Button>
    </Grid>
</UserControl>
```

من المهم ربط فضاء البرنامج للوصول الى الصنف patterns ضمن الوسم الرئيسي UserControl

او Window

```
xmlns:local="clr-namespace:My_First_Project"
```

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



ItemsSource="{Binding ElementName=UCDataTemplate1, Path=patterns}"

هنا ربطنا المصدر للقائمة مع برنامجنا الذي اسمه UCDataTemplate1 وبالتحديد الكائن patterns ضمن البرنامج.

وأخيراً يجب انشاء كائن من صنف العينات ضمن باني البرنامج الرئيسي:

```
public Patterns patterns{get;set;}
public DataTemplate1()
{
    patterns = new Patterns();
    InitializeComponent();
}
```

عند الضغط على الزر ضمن الواجهة سوف يتم إضافة عنصر للكائن patterns وتعديل الواجهة أوتوماتيكياً نتيجة الربط.

```
private void Button_Click_1(object sender, RoutedEventArgs e)
{
    patterns.Add(new Pattern(@"E:\stack.png", "StackOverFlow Arabi", 99));
}
```

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



المهندس فيصل الأسود

الشكال Shape

تأليف فيصل الأسود | مهندس برمجيات
يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



الاشكال Shape:

الاشكال هي عناصر هندسية يمكن التعامل معها كبقية العناصر ويمكنها الاستجابة للأحداث.

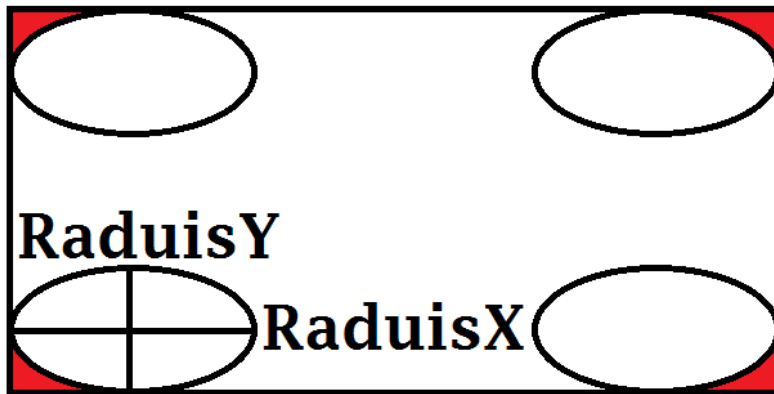
القطع الناقص: هو شكل قطع ناقص هندسي وما يلي بعض خصائصه الهامة:

الخاصة	الشرح
Width	عرض الشكل
Height	ارتفاع الشكل
Stroke	لون المحيط للشكل
Fill	لون الشكل
StrokeThickness	ثخانة المحيط

المستطيل: هو شكل هندسي له نفس خصائص القطع مع بعض الخصائص الأخرى:

الخاصة	الشرح
RaduisX	القطر الافقي للقطع الزاوي
RaduisY	القطر الشاقولي للقطع الزاوي

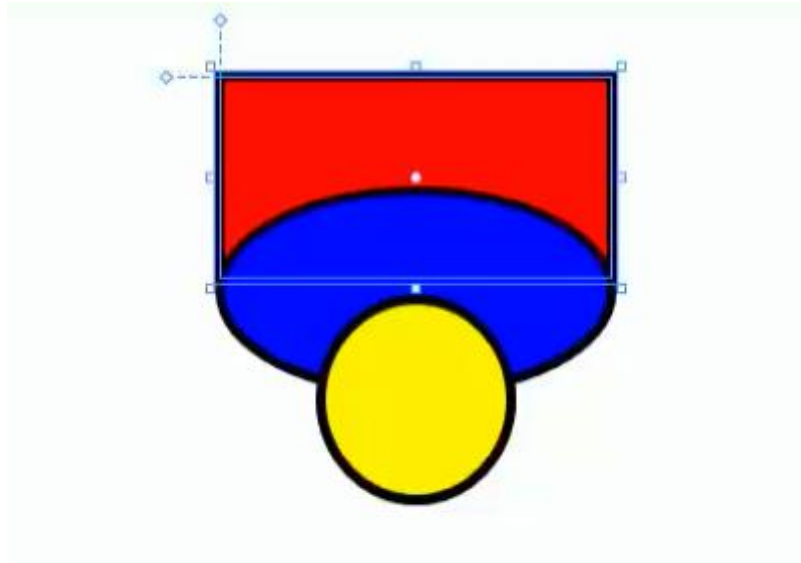
القطع الزاوي هو تقويس زوايا المستطيل على شكل قطع كما يلي:



تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

مثال: ليكن لدينا العناصر الهندسية التالية كما في الشكل:



اكتب البرنامج الذي يقوم برسم الاشكال السابقة مع مراعاة الترتيب بدءاً من الخلف (مستطيل – قطع ناقص – كرة).

```
<Window x:Class="My_First_Project.MainWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:local="clr-namespace:My_First_Project"
  Title="Win2" Height="500" Width="500" Loaded="Window_Loaded"
  Name="MyWindow" >

  <Canvas>
    <Rectangle Width="200" Height="100" Fill="Red" Stroke="Black"
      StrokeThickness="5" Canvas.ZIndex="1" Canvas.Left="204"
      Canvas.Top="48"></Rectangle>
    <Ellipse Width="200" Height="100" Fill="Blue" Stroke="Black"
      StrokeThickness="5" Canvas.ZIndex="2" Canvas.Left="254"
      Canvas.Top="91"/>
    <Ellipse Width="100" Height="100" Fill="Yellow" Stroke="Black"
      StrokeThickness="5" Canvas.ZIndex="3" Canvas.Left="195"
      Canvas.Top="114"/>
  </Canvas>
</Window>
```

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



<p>امكننا التحكم بترتيب العناصر عن طريق الخاصية <code>Canvas.ZIndex</code> وكلما كانت قيمتها للشكل اكبر كلما اقترب من الشاشة.</p>	<p>تنويه 1</p>
<p>في حال تساوي قيمة <code>Canvas.ZIndex</code> لعنصرين فإن العنصر المنشئ أخيراً يكون اقرب الى الشاشة.</p>	<p>تنويه 2</p>

المهندس فيصل الأسود

تأليف فيصل الأسود | مهندس برمجيات









يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



الخطوط Line:

هي خطوط مستقيمة يتم عن طريق تحديد نقطتين $(x1,y1)$ حتى $(x2,y2)$ ويحدد لون الخط بالخاصة Stroke كما تحدد ثخانتها عن طريق StrokeThickness.

لدينا للخطوط خصائص مميزة لرسم الأطراف:

StrokeStartLineCap		StrokeEndLineCap	
يحدد طرف الخط البدائي		يحدد طرف الخط النهائي	
			
Round	Flat	Round	Flat
			
Sequare	Triangle	Sequare	Triangle

يمكن التعامل مع المستقيم بكافة القيم الموجبة والسالبة وخارج المجال.

ملاحظة

مثال: الحالات التالية توضح حالة المستقيم.



داخل المجال



خارج المجال – القسم السالب



خارج المجال – القسم الموجب

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

يمكن التعامل مع المستقيم كأبي شكل هندسي قابل للتلوين والاستجابة للأحداث.

ملاحظة

مثال: ليكن لدينا الشكل التالي للمستقيم اكتب الكود الموافق.



والكود الموافق كما يلي:

```
<Window x:Class="Test.MainWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="MainWindow" Height="350" Width="525">
  <Grid>
    <Line X1="50" Y1="20" X2="200" Y2="200" StrokeThickness="25" >
      <Line.Stroke>
        <LinearGradientBrush StartPoint="0,0" EndPoint="1,1">
          <GradientStop Offset="0.1" Color="YellowGreen"></GradientStop>
          <GradientStop Offset="1" Color="Red"></GradientStop>
        </LinearGradientBrush>
      </Line.Stroke>
    </Line>
  </Grid>
</Window>
```

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



تقطيع الخطوط DashLine:

هي خصائص يمكن من خلالها رسم الخطوط على شكل خطوط مقطعة وبأشكال مختلفة، من هذه الخصائص نذكر:

الخاصة	الشرح	مثال
StrokeDashArray	مصفوفة تحدد شكل التقطيع	<p>StrokeDashArray="1 2"</p>
StrokeDashCap	تحدد شكل طرف التقطيع	<p>StrokeDashCap="Triangle"</p>
StrokeDashOffset	تحدد نقطة البداية من المصفوفة	<p>StrokeDashOffset="0"</p>

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

الخطوط المتعددة PolyLine:

هي عبارة عن سلسلة من الخطوط متتابعة بحيث تكون نهاية الأولى بداية الثانية.
من اهم خصائص الـ PolyLine نذكر:

الخاصة	الشرح
Points	<p>تحدد النقاط التي ستشكل هذا الخط المتعدد ويكون عدد المستقيمات يساوي عدد النقاط -1</p> <p>تكتب بالشكل العام:</p> <p>Points="x1 y1 x2 y2 x3 y3"</p> <p>او</p> <p>Points="x1,y1 x2,y2 x3,y3"</p>
StrokeLineJion	<p>وتحدد هذه الخاصة طريقة الانكسار بين المستقيمين ولها الحالات التالية:</p> <ul style="list-style-type: none"> • Bevel • Miter • Round 

مثال: اكتب الكود الموافق للشكل الظاهر كما يلي:

MainWindow



- □ ×

تأليف فيصل الأسود | مهندس برمجيات

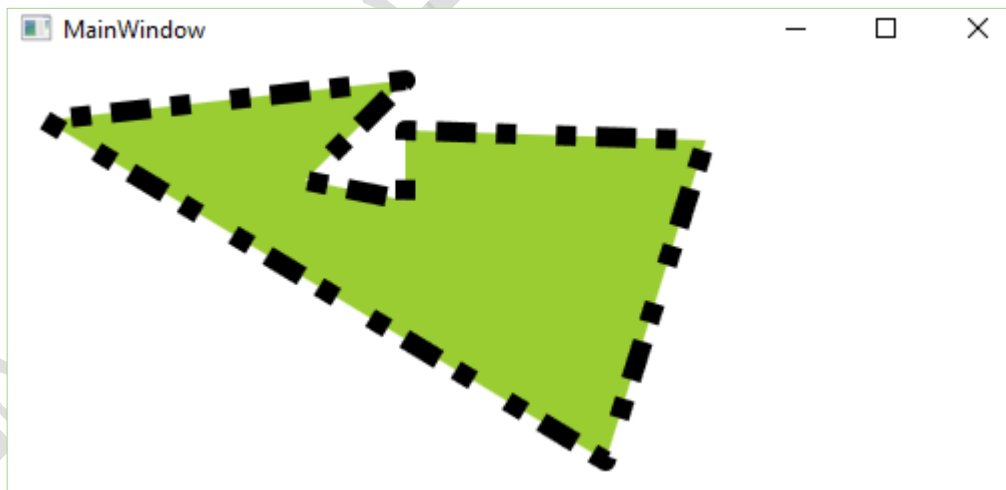
يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

```
<Window x:Class="Test.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="350" Width="525">
    <Grid>
        <Polyline Points="20 30 300 200 350 40" Stroke="Black"
        StrokeThickness="30" StrokeLineJoin="Round"></Polyline>
    </Grid>
</Window>
```

المضلعات Polygon: هي اشكال هندسية مكونة من اضلاع ويمكن ملأها بلون معين بحيث تكون مغلقة دوماً.

مايميز الـ Polygon عن الـ PolyLine هو إمكانية ملئ الأولى بلون معين وذلك كون المضلعات شكل مغلق.

مثال: اكتب الكود الموافق لرسم الشكل التالي:



```
<Grid>
    <Polygon Points="20 30 300 200 350 40 200 35 200 70 150 60 200 10"
            Fill="YellowGreen"
            StrokeDashArray="1 2 1"
            Stroke="Black"
            StrokeThickness="10"
            StrokeLineJoin="Round"></Polygon>
</Grid>
```

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

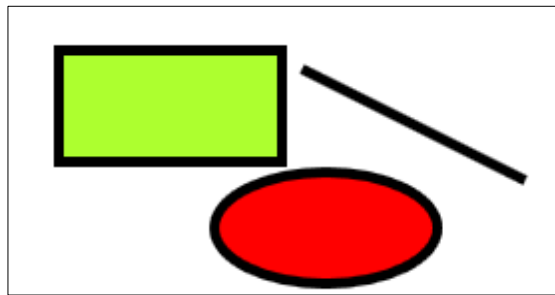
المسارات الهندسية Path:

هي مشابهة للأشكال Shape لكن يمكننا من بناء اشكال بمسارات معقدة وإمكانات اكبر.

لها نوعين:

- المسارات البسيطة (الأساسية): كما رأينا من الدروس السابقة كيف امكنا إنشاء قطع ناقص ودائرة وخط مباشرة ، وهذا يتحقق ايضاً ضمن المسارات الهندسية كأشكال جاهزة.

مثال: المطلوب انشاء الاشكال التالية باستخدام المسارات Path.



```
<Window x:Class="Test.MainWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="MainWindow" Height="350" Width="525">
  <Grid>
    <Path Fill="Red" Stroke="Black" StrokeThickness="5" Margin="228,209,172,51"
  Stretch="Fill">
      <Path.Data>
        <EllipseGeometry RadiusX="100" RadiusY="50"></EllipseGeometry>
      </Path.Data>
    </Path>

    <Path Fill="GreenYellow" Stroke="Black" StrokeThickness="5"
  Margin="150,150,250,110" Stretch="Fill">
      <Path.Data>
        <RectangleGeometry Rect="0 0 100 50"></RectangleGeometry>
      </Path.Data>
    </Path>

    <Path Stroke="Black" StrokeThickness="5" Margin="272,159,128,101"
  Stretch="Fill">
      <Path.Data>
        <LineGeometry StartPoint="0,0" EndPoint="100,100"></LineGeometry>
      </Path.Data>
    </Path>
  </Grid>
</Window>
```

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

المسارات المعقدة: هي شكل يتكون من مجموعة اشكال سابقة قد تكون بسيطة(أساسية) او معقدة،وهي تسهل ما قمنا به في مثالنا السابق بحيث اننا الان يمكننا انشاء كل الاشكال السابقة ضمن path واحد.

مثال ليكن لدينا هذا الشكل ،المطلوب رسمه باستخدام الـ Path.



```
<Path Stroke="Black" Fill="Red" StrokeThickness="5" Margin="185,124,215,136"
Stretch="Fill">
  <Path.Data>
    <GeometryGroup>
      <LineGeometry StartPoint="0,0"
EndPoint="100,100"></LineGeometry>
      <RectangleGeometry Rect="0 0 100 50"></RectangleGeometry>
      <EllipseGeometry RadiusX="100" RadiusY="50" ></EllipseGeometry>
    </GeometryGroup>
  </Path.Data>
</Path>
```

نلاحظ ان الاشكال الثلاثة ظهرت ضمن شكل واحد.

في المسارات Path عند تقاطع شكلين يتم اعتبار المساحة المتقاطعة بينهما فارغة ، ويمكن الغاء ذلك بكتابة الخاصة "FillRule="nonzero" ضمن الخصائص للـ GeometryGroup.

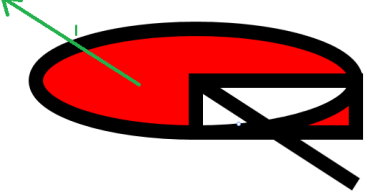
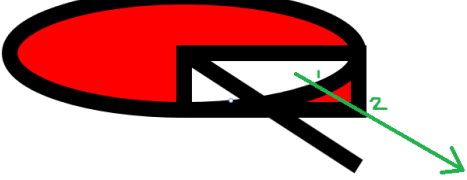
ملاحظة

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

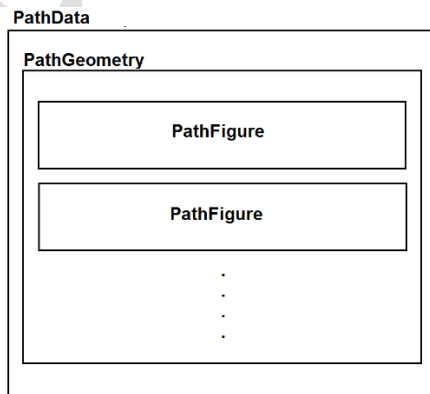
قاعدة الـEvenOdd:

تنص هذه القاعدة على ان أي نقطة ترسل شعاع الى اللانهاية وتحصل على تقاطعات للأشكال المغلقة بعدد زوجي لن يتم اظهارها ضمن الشكل تعتبر فراغ وما عدا ذلك الفردية يتم اظهارها او اعتبارها من الشكل.

	<p>تقاطعات فردية نلاحظ ان النقطة حصلت على تقاطع واحد فستظهر ضمن الشكل.</p>
	<p>تقاطعات زوجية نلاحظ ان النقطة حصلت على تقاطعين فلن تظهر ضمن الشكل.</p>

في مثالنا السابق شكلنا شكل معقد من اشكال أساسية او بسيطة، الان سوف نتطوع على مسارات معقدة اكثر بحيث يحدد كل نقطة منها بالتفصيل.

يجب ان نعلم ان الـ PathGeometry يمكن ان يتكون هو الاخر من اشكال جزئية معقدة كما في الهيكلية التالية:

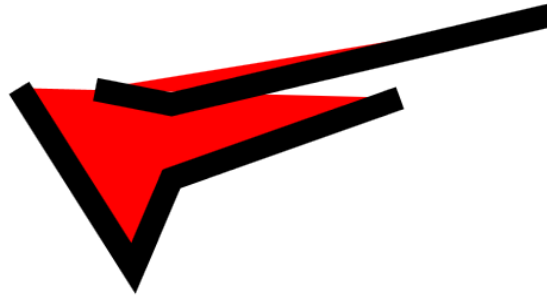


بحيث ان الشكل المعقد PathGeometry هو عبارة عن مجموعة اشكال معقدة PathFigure وكل شكل من نوع PathFigure يملك خطوطه ونقطة بدايته الخاصة.

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

مثال:



```
<Path Stroke="Black" Fill="Red" StrokeThickness="5" Margin="185,124,215,136"
Stretch="Fill">
  <Path.Data>
    <PathGeometry>
      <PathFigure StartPoint="0,0">
        <LineSegment Point="100 100"></LineSegment>
        <LineSegment Point="600 -500"></LineSegment>
      </PathFigure>
      <PathFigure StartPoint="-100,-10">
        <PolyLineSegment Points="50,1200 100,600 400 57"></PolyLineSegment>
      </PathFigure>
    </PathGeometry>
  </Path.Data>
</Path>
```

يمكن PathFigure ان يحوي عدد من الاشكال منها:

LineSegment	يأخذ نقطتين لرسم خط
PolylineSegment	يأخذ عدة نقاط مباشرة

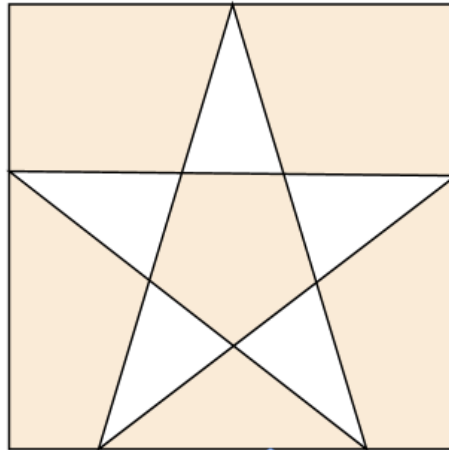
وله الخاصيتين:

IsFilled	تحدد هل الشكل ممتلئ
IsClosed	تحدد هل الشكل مغلق

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

مثال:



```

<Canvas>
  <Path Fill="AntiqueWhite" Stroke="Black" Canvas.Left="129" Canvas.Top="33">
    <Path.Data >
      <PathGeometry FillRule="Nonzero" >
        <PathFigure IsClosed="True" StartPoint="100,0">
          <LineSegment Point="160,200"></LineSegment>
          <LineSegment Point="0,75"></LineSegment>
          <LineSegment Point="200,77"></LineSegment>
          <LineSegment Point="40,200"></LineSegment>
        </PathFigure>
        <PathFigure StartPoint="0,0" IsClosed="True">
          <PolyLineSegment Points="0,200 200,200 200,0"></PolyLineSegment>
        </PathFigure>
      </PathGeometry>
    </Path.Data>
  </Path>
</Canvas>

```

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

المسارات المنحنية (المنحنيات):

المسارات المنحنية هي عبارة عن مسارات تمتلك انحناء او تقوس ضمن مسارها ومن اهم خصائصها نذكر:

الخاصة	الشرح
Points	تحدد النقاط التي يمر منها المنحني.
SweepDirect	تحدد اتجاه سير المنحني باتجاه عقارب الساعة ClockWise او عكس اتجاه عقارب الساعة .
RotationAngle	تحدد ميل زاوية الانعطاف عن النقطتين المشكلتين للخط الجزئي.
IsLargeAec	تحدد هل المنحني يصل فقط النقاط او يصل النقاط ويأخذ حجم معين.
Size	تحدد حجم المنحني في الواجهة.
IsFilled	هل المنحني ملون
IsClosed	هل المنحني مغلق

مثال: أنشئ Arc كما في الشكل



```
<Canvas>
  <Path Fill="Azure" Stroke="Black" StrokeThickness="5" Canvas.Left="73"
Canvas.Top="124">
  <Path.Data>
    <PathGeometry>
      <PathFigure StartPoint="100,100" IsClosed="False" IsFilled="False" >
        <ArcSegment Point="200,100" SweepDirection="Clockwise"
          RotationAngle="0"
          IsLargeArc="True"
          Size="150,30"
        >>/ArcSegment>
      </PathFigure>
    </PathGeometry>
  </Path.Data>
</Path>
</Canvas>
```

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

مثال: ارسم الشكل.



```
<Canvas>
  <Path Fill="Azure" Stroke="Black" StrokeThickness="5" Canvas.Left="73"
Canvas.Top="124">
    <Path.Data>
      <PathGeometry>
        <PathFigure StartPoint="0,0" IsClosed="False" IsFilled="False" >
          <BezierSegment Point1="50,0" Point2="100,100"
Point3="150,50"></BezierSegment>
        </PathFigure>
      </PathGeometry>
    </Path.Data>
  </Path>
</Canvas>
```

لغة Geometry Mini Language:

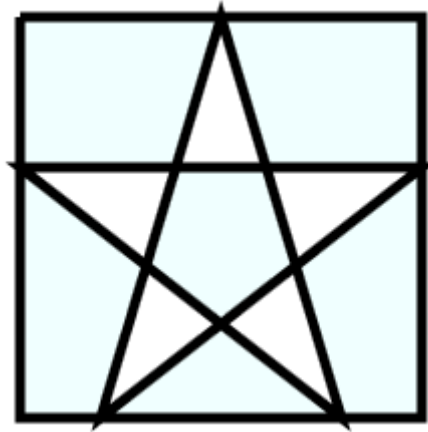
هي لغة مكونة من مجموعة من الرموز قام بابتكارها مطورو مايكروسوفت بهدف تسهيل توصيف الرسومات.

الرمز	الشرح
M	بداية مستقيم يليها النقطة مباشرة
L	مستقيم تحدد نقطة نهايته
Z	نهاية القطعة
H	خط افقي يليه الـ X الخاصة بالقطعة .
V	خط عامودي يليه الـ Y الخاصة بالقطعة.

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

مثال:



```
<Canvas>
<Path Fill="Azure" Stroke="Black" StrokeThickness="5" Canvas.Left="73"
Canvas.Top="124"
Data="M100,0 L160,200 L0,75 L200,75 L40,200 Z M0,0 H200 v200 H0 V0"
>
</Path>
</Canvas>
```

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

التلوين والفرش Brushes

المهندس فيصل الأسود
يطلق الأسود

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



التلوين Brushes:

تلوين العناصر ضمن WPF له الطرق الأساسية لاربعة التي سنذكرها بالتفصيل:

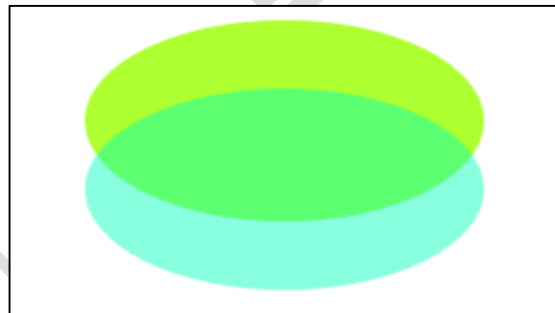
- اللون الواحد Solid Color Brush: هو عبارة عن لون واحد مصمت من نوع RGB ويمكن إعطائه القيمة بطريقتين:

Color="Black"	○ اسم اللون مباشرة.
Color="#ff66ab"	○ طريقة الـ RGB.

كما يوجد خاصية هي الـ Opacity التي تساعد على تغيير قيمة شفافية اللون ، وتعطى ضمن نظام الـ RGB بالشكل:

Opacity	R	G	B
---------	---	---	---

مثال:



```
<Grid>
  <Ellipse Width="200" Height="100" Margin="146,256,146,113" >
    <Ellipse.Fill>
      <SolidColorBrush Color="GreenYellow"></SolidColorBrush>
    </Ellipse.Fill>
  </Ellipse>

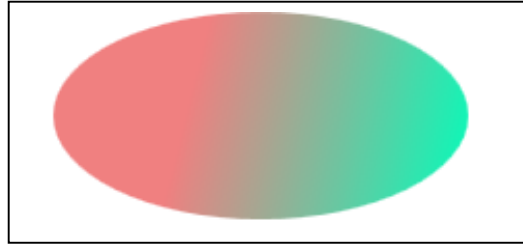
  <Ellipse Width="200" Height="100" Margin="146,290,146,79">
    <Ellipse.Fill>
      <SolidColorBrush Color="#7700ffbb"></SolidColorBrush>
    </Ellipse.Fill>
  </Ellipse>
</Grid>
```

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

- التلوين بتدرج خطي Linear Gradient Brush: هو تدرج خطي يبدأ من نقطة حتى نقطة أخرى على شكل مستقيم.

- مثال:



```
<Grid>
  <Ellipse Width="200" Height="100" Margin="137,66,155,303" >
    <Ellipse.Fill>
      <LinearGradientBrush StartPoint="0,0.5" EndPoint="1,1" >
        <GradientStop Color="LightCoral" Offset="0.3"></GradientStop>
        <GradientStop Color="#00ffbb" Offset="1"></GradientStop>
      </LinearGradientBrush>
    </Ellipse.Fill>
  </Ellipse>
  <TextBlock FontSize="100" Margin="118,224,-118,-224" Text="TEXR">
    <TextBlock.Foreground>
      <LinearGradientBrush StartPoint="0,0.5" EndPoint="0.1,0.5"
SpreadMethod="Repeat">
        <GradientStop Color="MediumAquamarine" Offset="0.3"></GradientStop>
        <GradientStop Color="#00ffbb" Offset="1"></GradientStop>
      </LinearGradientBrush>
    </TextBlock.Foreground>
  </TextBlock>
</Grid>
```

ملاحظة الـ Offset تحدد القيمة التي سيحتلها اللون كجزء من مسار المستقيم وقيمة عشرية.

ملاحظة

في حال قيمة الـ Offset لم تكتمل حتى نهاية المستقيم ومنه يمكن استخدام احدى الطرق لاكمال اللون : Pad – Reflect – Repeat

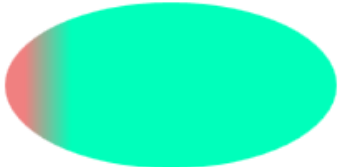


ملاحظة

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



سوف نذكر حالياً مع الشرح الخاصة SpreadMethod والتي تعمل على اكمال اللون ولها الحالات الثلاث:

Pad	Reflect	Repeat
اكمال اللون الأخير حتى النهاية	اكمال اللون بعكس الألوان حتى النهاية	تكرار التدرج نفسه حتى النهاية
		

مثال:

TEXR

```
<TextBlock FontSize="100" Margin="118,224,-118,-224" Text="TEXR">
  <TextBlock.Foreground>
    <LinearGradientBrush StartPoint="0,0.5" EndPoint="0.1,0.5" SpreadMethod="Repeat">
      <GradientStop Color="MediumAquamarine" Offset="0.3"></GradientStop>
      <GradientStop Color="#00ffbb" Offset="1"></GradientStop>
    </LinearGradientBrush>
  </TextBlock.Foreground>
</TextBlock>
```

تأليف فيصل الأسود | مهندس برمجيات

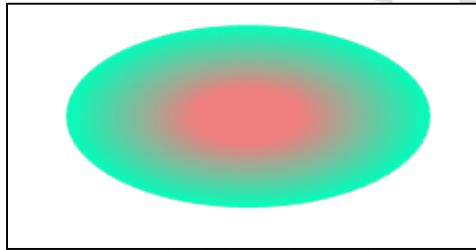
يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

- التلوين بتدرج الدائري Radial Gradient Brush: هي تلوين متدرج بشكل دوائر متداخلة ، ويمكن تحديد مركز هذه الدوائر.

من خواص التدرج الدائري:

الخاصة	الشرح
RadiusX	قطر التدرج الداخلي افقياً.
RadiusY	قطر التدرج الداخلي عامودياً.
spreadMethod	طريقة اكمال اللون.
GradientOrigin	مركز الدوائر ويكون قيمة نسبية من 0 <--- 1

مثال:



```
<Grid>
  <Ellipse Width="200" Height="100" Margin="137,66,155,303" >
    <Ellipse.Fill>
      <RadialGradientBrush
        GradientOrigin="0.5,0.5"
        RadiusX="0.5"
        RadiusY="0.5"
        SpreadMethod="Repeat">
        <GradientStop Color="LightCoral" Offset="0.3"></GradientStop>
        <GradientStop Color="#00ffbb" Offset="1"></GradientStop>
      </RadialGradientBrush>
    </Ellipse.Fill>
  </Ellipse>
</Grid>
```

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

التلوين بالصور Image Brush: هو إعطاء العنصر صورة لتمثل لونه.

مثال:



```
<Grid>
  <Ellipse Width="200" Margin="135,86,157,191" >
    <Ellipse.Fill>
      <ImageBrush ImageSource="D:\5 logo 2.png"
        AlignmentX="Right"
        AlignmentY="Center"
        Stretch="Fill"></ImageBrush>
    </Ellipse.Fill>
  </Ellipse>
</Grid>
```

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

الخاصية Opacity Mask:

هي خاصية تمكن من التحكم بالشفافية للعنصر بشكل متدرج.

ولفهمها بشكل أوسع نقول انها تعطي جميع الألوان ثقلها ما عدا اللون Transparent يظهر بشكل شفاف.

مثال:



```
<Grid>
  <Image Source="e:\Stack.png"></Image>
  <Ellipse Width="200" Margin="130,252,162,46">
    <Ellipse.Fill>
      <ImageBrush ImageSource="D:\5 logo 2.png"></ImageBrush>
    </Ellipse.Fill>
    <Ellipse.OpacityMask>
      <RadialGradientBrush >
        <GradientStop Color="Aquamarine" Offset="0.2"></GradientStop>
        <GradientStop Color="Aquamarine" Offset="0.3"></GradientStop>
        <GradientStop Color="Transparent" Offset="1"></GradientStop>
      </RadialGradientBrush>
    </Ellipse.OpacityMask>
  </Ellipse>
  <Button Panel.ZIndex="2" Content="Button" FontSize="20" Background="Aqua"
Margin="162,252,198,129">
    <Button.OpacityMask>
      <LinearGradientBrush StartPoint="0.5,0" EndPoint="0.5,1">
        <GradientStop Color="Red" Offset="0.1"/>
        <GradientStop Color="Transparent" Offset="1"/>
      </LinearGradientBrush>
    </Button.OpacityMask>
  </Button>
</Grid>
```

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

التحويلات الهندسية Transforms

المهندس فيصل الأسود
يطلق الأسود

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



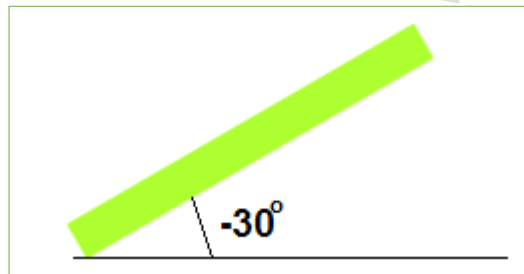
التحويلات الهندسية Transforms

هي تحويلات يمكننا من اجراء عمليات مثل (تدوير - نقل - إمالة - تقييس) للعناصر.

- التدوير Rotation: نذكر اهم خصائصه:

الخاصة	الشرح
Angle	قيمة الزاوية المطلوب تدوير الشكل فيها
CenterX	الاحداثيات الافقية لمركز تدوير الشكل
CenterY	الاحداثيات العمودية لمركز تدوير الشكل

مثال:



```
<Path Fill="GreenYellow"
  Data="{StaticResource RectResource}" Canvas.Left="98" Canvas.Top="18">
  <Path.RenderTransform>
    <RotateTransform CenterX="200" CenterY="20"
  Angle="30"></RotateTransform>
  </Path.RenderTransform>
</Path>
```

```
<UserControl.Resources >
  <GeometryGroup x:Key="RectResource">
    <RectangleGeometry Rect="0 0 200 20"></RectangleGeometry>
  </GeometryGroup>
</UserControl.Resources >
```

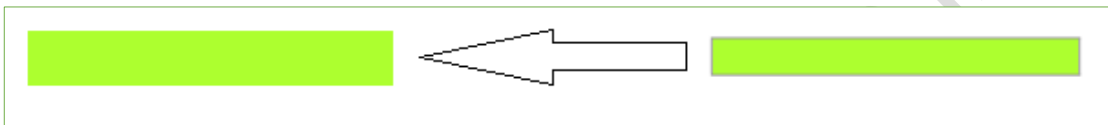
تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

التقييس Scaling:

الخاصية	الشرح
ScaleX	قيمة التقييس افقياً (اصغر من الواحد = تصغير الشكل افقياً)
ScaleY	قيمة التقييس عامودياً (اصغر من الواحد = تصغير الشكل عامودياً)
CenterX	احداثيات مركز التقييس افقياً
CenterY	احداثيات مركز التقييس عامودياً

مثال:



```
<Path Fill="GreenYellow" Data="{StaticResource RectResource}" Canvas.Left="98"
Canvas.Top="18">
  <Path.RenderTransform>
    <ScaleTransform CenterX="0" CenterY="0" ScaleY="1"></ScaleTransform>
  </Path.RenderTransform>
</Path>
```

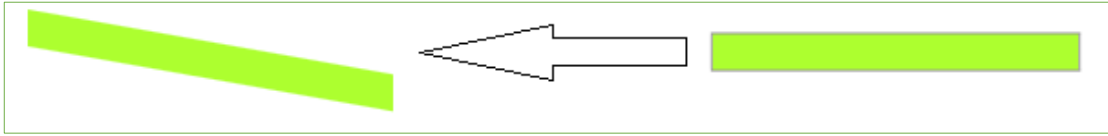
الإمالة Skew:

الخاصية	الشرح
AngleX	قيمة الإمالة افقياً
AngleY	قيمة الإمالة عامودياً
CenterX	احداثيات مركز الإمالة افقياً
CenterY	احداثيات مركز الإمالة عامودياً

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

مثال:



```
<Path Fill="GreenYellow"
      Data="{StaticResource RectResource}" Canvas.Left="98" Canvas.Top="18">
  <Path.RenderTransform>
    <SkewTransform CenterX="0" CenterY="0" AngleX="0"
  AngleY="10"></SkewTransform>
  </Path.RenderTransform>
</Path>
```

النقل Translation: هي تحريك مركز الجسم من نقطة لنقطة أخرى.

مثال: نقل العنصر للنقطة (50,20).

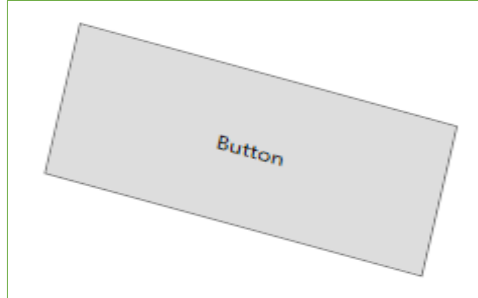
```
<Path Fill="AntiqueWhite"
      Data="{StaticResource RectResource}" Canvas.Left="108"
  Canvas.Top="280">
  <Path.RenderTransform>
    <TranslateTransform X="50" Y="20"></TranslateTransform>
  </Path.RenderTransform>
</Path>
```

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

مجموعة التحويلات: يمكن ان يطبق مجموعة من التحويلات على شكل معين.

مثال:



```
<Button Width="200" Height="100" Content="Button" Canvas.Left="455" Canvas.Top="160">
  <Button.RenderTransform>
    <TransformGroup>
      <SkewTransform AngleX="10"></SkewTransform>
      <RotateTransform Angle="20"></RotateTransform>
    </TransformGroup>
  </Button.RenderTransform>
</Button>
```

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube

الحركة Animation

المهندس فيصل الأسود

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



الحركة Animation:

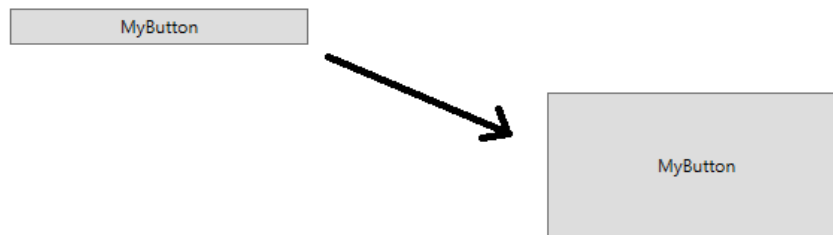
هي تحريك الكائنات ضمن فضاء الواجهة بشكل ثنائي او ثلاثي البعد ، وذلك يكون غالباً نتيجة تنفيذ احداث مسببة لتلك الحركة.

توضع الحركات ضمن ما يسمى بالقوادح أو Triggers ويحدد ضمنها الاحداث التي ستستجيب لتلك القوادح.

يوجد عدة أنواع للحركة منها:

- الحركة المضاعفة DoubleAnimation : غالباً ما يستخدم في تحريك او تغيير مقاسات العناصر.

مثال: أنشئ زر يتحرك من النقطة 100 إلى النقطة 300 افقياً خلال 5 ثانية مع زيادة ارتفاعه من 25 إلى 100 خلال 2 ثانية وذلك عند الضغط عليه (تنفيذ الحدث Button.Click).



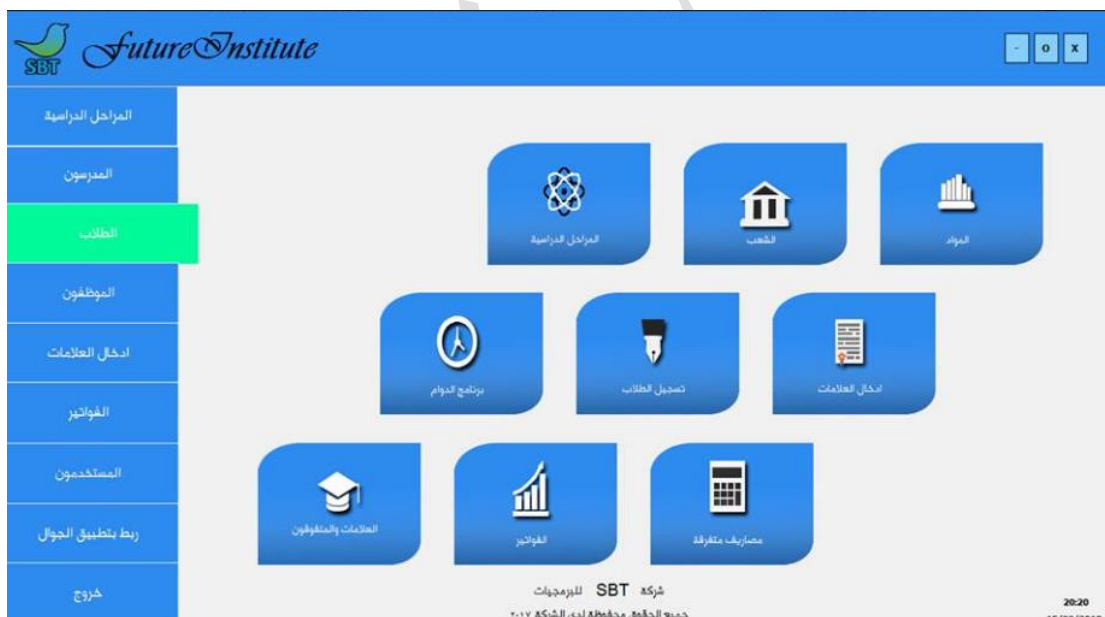
```
<Canvas>
<Button Height="25" Width="200" Canvas.Left="55" Canvas.Top="84" Name="MyBtn"
Content="MyButton">
  <Button.Triggers>
    <EventTrigger RoutedEvent="Button.Click">
      <BeginStoryboard>
        <Storyboard >
          <DoubleAnimation Storyboard.TargetName="MyBtn"
Storyboard.TargetProperty="(Canvas.Left)"
From="100" To="300" Duration="0:0:5">
          </DoubleAnimation>
          <DoubleAnimation Storyboard.TargetName="MyBtn"
Storyboard.TargetProperty="Height"
From="25" To="100" Duration="0:0:2">
          </DoubleAnimation>
        </Storyboard>
      </BeginStoryboard>
    </EventTrigger>
  </Button.Triggers>
</Button>
</Canvas>
```

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



تصميم كامل:



يمكنك متابعة طريقة التصميم كاملة على القناة StackOver flow Arabi

تأليف فيصل الأسود | مهندس برمجيات

يمكنك الحصول على شرح لكامل محتويات الكتاب على قناة الـ Youtube



مقدمة الى WPF:

هل نحتاج الى واجهات مستخدم UI أخرى بديلة عن الواجهات الحالية Windows Form التي لا تكون عادة بمظهر جيد. جميعنا نعلم ان التطبيقات الجديدة تحتاج الى طرق جديدة لإظهار المعلومات بدلا من واجهة مستطيلة. يكون هنالك اشكال صور مخططات والوان.

تقنيات الرسومات الموجودة حاليا تجعل من الصعب خلق تطبيقات مرئية وكذلك حتى الان العتاد الخاص بالرسومات من كروت الشاشة وحوافظ الذاكرة RAM مازالت قاصرة حتى عن معالجة بعض تطبيقات Windows Form العادية. يمكنك خلق واجهة من صور متحركة لكن ذلك يحتاج الى جهد كبير وامتلاكنا مجموعة من الأدوات الخاصة لذلك. واستخدام رسومات ثلاثية البعد 3D تعد مستحيلة بدون استخدام مكتبات إضافية حيث ان مكتبات الـ GDI الأساسية المدعومة من Windows مازالت ضعيفة.

