Pengantar XAML

XAML adalah singkatan dari Extensible Application Markup Language. Ini bahasa yang sederhana berdasarkan XML untuk membuat dan menginisialisasi objek .NET dengan hubungan hirarkis. Meskipun awalnya diciptakan untuk WPF dapat digunakan untuk membuat segala jenis objek.

Perkembangannya XAML digunakan untuk membuat antarmuka pengguna di WPF, Silverlight, menyatakan alur kerja di WF dan untuk kertas elektronik dalam standar XPS.

Semua kelas di WPF memiliki konstruktor tanpa parameter dan memiliki banyak properti. Itu dilakukan untuk membuat WPF sangat cocok untuk bahasa XML seperti XAML.

Keuntungan XAML

Semua yang dapat Anda lakukan dalam XAML juga dapat dilakukan dalam kode. XAML hanyalah cara lain untuk membuat dan menginisialisasi objek (dalam hal ini antarmuka aplikasi). Anda dapat menggunakan WPF tanpa menggunakan XAML. Programmer dibebaskan untuk memilih jika Anda ingin mendeklarasikannya dalam XAML atau menuliskannya dalam kode (untuk komponen UI. Menyatakan UI Anda di XAML memiliki beberapa keunggulan:

- Kode XAML pendek dan jelas untuk dibaca
- Pemisahan kode dan logika desainer
- Alat desain grafis seperti Expression Blend membutuhkan XAML sebagai sumber.
- Pemisahan logika XAML dan UI memungkinkannya untuk dengan jelas memisahkan peran perancang dan pengembang.

XAML vs Kode

Sebagai contoh, kita membangun StackPanel sederhana dengan blok teks dan tombol di XAML dan membandingkannya dengan kode yang sama di C #.

```
<StackPanel>
    <TextBlock Margin="20">Welcome to the World of XAML</TextBlock>
    <Button Margin="10" HorizontalAlignment="Right">OK</Button>
</StackPanel>
```

Hal yang sama dinyatakan dalam C # akan terlihat seperti ini:

```
// Create the StackPanel
StackPanel stackPanel = new StackPanel();
```

```
this.Content = stackPanel;

// Create the TextBlock
TextBlock textBlock = new TextBlock();
textBlock.Margin = new Thickness(10);
textBlock.Text = "Welcome to the World of XAML";
stackPanel.Children.Add(textBlock);

// Create the Button
Button button = new Button();
button.Margin= new Thickness(20);
button.Content = "OK";
stackPanel.Children.Add(button);
```

Seperti yang Anda lihat adalah versi XAML jauh lebih pendek dan lebih jelas untuk dibaca. Dan itulah kekuatan ekspresi XAML.

Properti sebagai Elemen

Properti biasanya ditulis sebaris yang dikenal dari XML `<Button Content="OK" />` . Tetapi bagaimana jika kita ingin meletakkan objek yang lebih kompleks sebagai konten seperti gambar (yang mana objek yang ditambahkan mungkin memiliki properti sendiri). Untuk melakukan itu kita bisa menggunakan sintaks elemen properti. Ini memungkinkan mengekstrak properti sebagai elemen baru yang merupakan elemen anak.

```
<Button>
  <Button.Content>
     <Image Source="Images/OK.png" Width="50" Height="50" />
  </Button.Content>
</Button>
```

Ekstensi Markup

Ekstensi markup adalah placeholder dinamis untuk nilai atribut di XAML. Mereka menyelesaikan nilai properti saat runtime. Ekstensi markup ditandai dengan oleh kurung kurawal (Contoh: `Background="{StaticResource NormalBackgroundBrush}"`). WPF memiliki beberapa ekstensi markup `MarkupExtension` , tetapi Anda dapat menulis ekstensi Anda sendiri, yang berasal dari `MarkupExtension` . Berikut adalah ekstensi markup bawaan:

- **Binding**

  Untuk mengikat nilai dua properti bersama.
- **StaticResource**

  Satu kali pencarian entri sumber daya
- **DynamicResource**

Memperbarui pencarian entri sumber daya secara otomatis

- **Template Binding**

  Untuk mengikat properti templat kontrol ke properti dependensi kontrol

- **x: Static**]

  Memecahkan nilai properti statis.

- **x: Null**
- Kembali `null`

Identifier pertama dalam sepasang kurung kurawal adalah nama ekstensi. Semua pengidentifikasi sebelumnya diberi nama parameter dalam bentuk Properti = Nilai. Contoh berikut menunjukkan label yang `Content` terikat dengan `Text` dari `textBox`. Saat Anda mengetik teks ke dalam `textBox`, properti teks berubah dan ekstensi markup yang mengikat secara otomatis memperbarui konten label.

```
<TextBox x: Name = "textBox" />
<Label Konten = "{Teks Jilid, ElementName = textBox}" />
```

Namespace

Di awal setiap file XAML Anda harus menyertakan dua ruang nama.
Yang pertama adalah `http://schemas.microsoft.com/winfx/2006/xaml/presentation` . Itu dipetakan ke semua kontrol WPF di `System.Windows.Controls` .
Yang kedua adalah `http://schemas.microsoft.com/winfx/2006/xaml` yang dipetakan ke `System.Windows.Markup` yang menentukan kata kunci XAML.
Pemetaan antara namespace XML dan namespace CLR dilakukan oleh atribut `XmlnsDefinition` di tingkat perakitan. Anda juga bisa langsung memasukkan namespace CLR di XAML dengan menggunakan awalan `clr-namespace:`

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

Ringkasan XAML

XAML adalah bahasa pemrograman deskriptif baru yang dikembangkan oleh Microsoft untuk menulis antarmuka pengguna untuk aplikasi yang dikelola generasi berikutnya. XAML adalah bahasa untuk membangun antarmuka pengguna untuk aplikasi Windows dan Mobile yang menggunakan Windows Presentation Foundation (WPF), UWP, dan Xamarin Forms.

Tujuan XAML sederhana, untuk membuat antarmuka pengguna menggunakan bahasa markup yang terlihat seperti XML. XAML menggunakan format XML untuk elemen dan atribut. Setiap elemen dalam XAML mewakili objek yang merupakan instance dari suatu tipe. Ruang lingkup tipe (kelas, enumerasi, dll.) Didefinisikan sebagai namespace yang secara fisik berada di assembly (DLL) dari pustaka .NET Framework.

Mirip dengan XML, sintaks elemen XAML selalu dimulai dengan braket sudut pembuka (**<**) dan diakhiri dengan braket sudut penutup (**>**). Setiap tag elemen juga memiliki tag awal dan tag akhir. Misalnya, objek Tombol diwakili oleh elemen objek **`<Button>.`** Potongan kode berikut mewakili elemen objek Button.

```
<Button>/ <Button>
```

Atau, Anda dapat menggunakan format self-closing untuk menutup braket.

```
<Button>.
```

Elemen objek di XAML mewakili tipe. Tipe bisa berupa komponen control, kelas atau objek lain yang didefinisikan dalam pustaka(library) Framework.

Elemen Root

Setiap dokumen XAML harus memiliki elemen root. Elemen root biasanya berfungsi sebagai container dan mendefinisikan ruang nama dan properti dasar elemen. Tiga elemen root paling umum adalah **`<Windows/>`**, **`<Page/>`**, dan **`<UserControl>`**. **`<ResourceDirectory />`** dan **`<Application />`** adalah dua elemen root lainnya yang dapat digunakan dalam file XAML.

Elemen Window mewakili container Window. Potongan kode berikut menunjukkan elemen Window **`Height`**, **`Width`**, **`Title`** dan atribut **`x:Name`**. Atribut **`x:Name`** dari suatu elemen mewakili ID dari suatu elemen yang digunakan untuk mengakses elemen dalam program(dalam hal ini C#). Potongan kode juga menetapkan atribut **`xmlns`** dan **`xmlns:x`**

yang mewakili ruang nama yang digunakan dalam kode. Atribut **`x:Class`** mewakili kode dalam nama class .

```
<Window x:Class="HelloXAML.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">
</Window>
```

Elemen **`<Page/>`**, mewakili container Page/halaman. Potongan kode berikut membuat container page. Kode juga menetapkan atribut FlowDirection yang mewakili layout dari konten halaman.

```
<Page
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    x:Class="WPFApp.Page1"
    x:Name="Page"
    WindowTitle="Page"
    FlowDirection="LeftToRight"
    Width="640" Height="480"
    WindowWidth="640" WindowHeight="480">
</Page>
```

Elemen <**`UserControl>`**. mewakili container kontrol pengguna. Potongan kode berikut mewakili container kontrol pengguna.

```
<UserControl x:Class="HelloXAML.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
</UserControl>
```

Namespace XAML

Bagian dari elemen root masing-masing XAML adalah dua atau lebih atribut yang memiliki prefix(awalan) xmlns dan xmlns:x.

```
< Window
```

```
2. xmlns = "http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3. xmlns: x = "http://schemas.microsoft.com/winfx/2006/xaml"
4. …… >
5. </Window>
```

Atribut xmlns menunjukkan namespace XAML default sehingga elemen objek yang digunakan dalam XAML dapat ditentukan tanpa awalan. Selain itu, awalan x: digunakan dengan lebih dari sekedar namespace. Berikut adalah beberapa sintaks awalan x: yang digunakan dalam XAML.

1. **x: Key**: Menetapkan kunci unik untuk setiap sumber daya di ResourceDictionary.
2. **x: Class**: Nama kelas memberikan kode-belakang untuk halaman XAML.
3. **x: Name**: Nama objek run-time unik untuk instance yang ada dalam kode run-time setelah elemen objek diproses.
4. **x: Static**: Mengaktifkan referensi yang mengembalikan nilai statis yang bukan properti yang kompatibel dengan XAML.
5. **x: Type**: Membangun referensi Tipe berdasarkan nama tipe.

Elemen dan Atribut

Jenis WPF atau Windows RT diwakili oleh elemen XAML. Elemen **`<Page>`** dan **`<Button>`** mewakili masing-masing halaman dan tombol kontrol. Elemen Tombol XAML yang tercantum dalam kode berikut mewakili kontrol tombol.

```
<Button/>
```

Setiap elemen seperti **`<Page>`** atau **`<Button>`** memiliki atribut yang dapat diatur dalam elemen itu sendiri. Atribut suatu elemen mewakili properti dari tipe tersebut. Misalnya, Tombol memiliki properti Tinggi, Lebar, Latar Belakang, dan Latar depan yang masing-masing mewakili tinggi, lebar, warna latar depan, dan warna latar belakang tombol. Properti Konten Tombol mewakili teks kontrol tombol. Properti **`x: Name`** mewakili ID unik dari kontrol yang dapat digunakan untuk mengakses kontrol dalam kode di belakang.

Potongan kode berikut menetapkan ID, tinggi, lebar, warna latar belakang, warna latar depan, nama dan ukuran font serta konten kontrol tombol.

```
1. <Button Content="Click Me" Width="200" Height="50"
2.     Background="Orange" Foreground="Blue"
3.     FontSize="20" FontFamily="Georgia" FontWeight="Bold"
4.     x:Name="ClickButton">
5. </Button>
```

Properti Konten

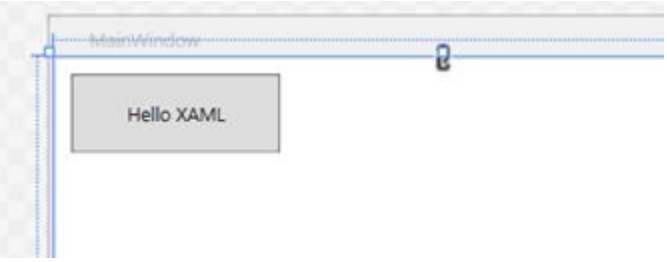
Setiap elemen objek XAML mampu menampilkan berbagai jenis konten. XAML menyediakan properti khusus yang disebut Konten yang berfungsi untuk menampilkan konten elemen tergantung pada kemampuan elemen. Misalnya, properti Konten dari Tombol dapat menjadi set ke string, objek, UIElement, atau event dan container. Namun, properti Konten ListBox diatur menggunakan properti Item.

Catatan: Beberapa elemen objek XAML mungkin tidak memiliki properti Konten yang tersedia secara langsung. Itu harus diatur melalui properti.

Potongan kode berikut ini membuat Kontrol tombol dan menetapkan properti Kontennya ke string "Halo XAML".

```
1.
   <Button Height="50" Margin="10,10,350,310" Content="Hello XAML" />
2.
      //Here is an alternative way to set the Content property of a B
      utton.
3. <Button Height="50" Margin="10,10,350,310">Hello XAML</Button>
```

Output dari kode di atas terlihat seperti berikut.



Elemen tombol dapat menampilkan elemen anak(child) lainnya sebagai kontennya. Kode berikut yang tercantum menetapkan elemen Rectangle sebagai konten Button.

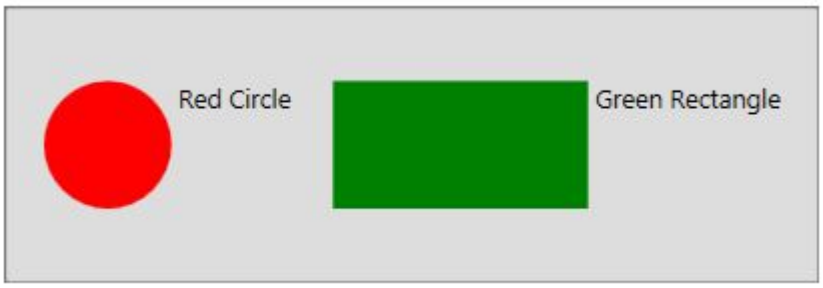
```
1. <Button Heigh ="80" Margin="10,80,300,170">
2.    <Rectangle Heigh="60" Width="120" Content="Green"/>
3. </Button>
```

Outputnya terlihat seperti berikut.

Properti konten juga dapat berupa container atau elemen induk yang menampung elemen anak. Kode yang tercantum di bawah ini menetapkan container StackPanel dengan 5 elemen anak sebagai konten Button.

```
1. <Button Margin="10,201,100,40">
2.     <StackPanel Orientation="Horizontal">
3.         <Ellipse Height="60" Width="60" Fill="Red"/>
4.
           <TextBlock TextAlignment="Center"><Run Text=" Red Circle"/
           ></TextBlock>
5.
           <TextBlock TextAlignment="Center"><Run Text=" "/></TextBl
           ock>
6.
           <Rectangle Height="60" Width="120" Fill="Green"></Rectang
           le>
7.
           <TextBlock TextAlignment="Center"><Run Text=" Green Recta
           ngle"/></TextBlock>
8.     </StackPanel>
9. </Button>
```

Ini adalah output yang baru.



Berikut adalah kode XAML lengkap.

```
1. <Window x:Class="ContentPropertySample.MainWindow"
2.
   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3. xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4. Title="MainWindow" Height="400" Width="500" >
5. <Grid x:Name="ParentGrid">
6. <Button Height="50" Margin="10,10,350,310" Content="Hello XAML"/>
7. <Button Height="80" Margin="10,80,300,170">
8. <Rectangle Height="60" Width="120" Fill="Green"></Rectangle>
9. </Button>
10.     <Button Margin="10,201,100,40">
11.     <StackPanel Orientation="Horizontal">
12.     <Ellipse Height="60" Width="60" Fill="Red"/>
13.
        <TextBlock TextAlignment="Center"><Run Text=" Red Circle"/><
        /TextBlock>
14.     <TextBlock TextAlignment="Center"><Run Text=" "/></TextBlock>
15.
   <Rectangle Height="60" Width="120" Fill="Green"></Rectangle>
16.
        <TextBlock TextAlignment="Center"><Run Text=" Green Rectangl
        e"/></TextBlock>
17.     </StackPanel>
18.     </Button>
19.     </Grid>
20.     </Window>
21.     </Button>
22.     </Grid>
23.     </Window>
```

Hasil akhir.

Seperti yang dapat Anda bayangkan dari contoh-contoh sebelumnya, Anda dapat meng-host antarmuka pengguna dengan isi elemen XAML.

Kode berikut membuat Kontrol tombol sebelumnya secara dinamis dalam kode dan mengatur properti Konten mereka masing-masing ke string, Rectangle dan StackPanel.

```
1. // Button with string content
2. Button helloButton = new Button();
3. helloButton.Margin = new Thickness(10,10,350,310);
4. helloButton.Content = "Hello XAML";
5. // Button with a UIElement
6. Button buttonWithRectangle = new Button();
7. buttonWithRectangle.Height = 80;
8. buttonWithRectangle.Margin = new Thickness(10, 80, 300, 170);
9. // Create a Rectangle
10.     Rectangle greenRectangle = new Rectangle();
11.     greenRectangle.Height = 60;
12.     greenRectangle.Width = 120;
13.     greenRectangle.Fill = Brushes.Green;
14.     // Set Rectangle as Button.Content
15.     buttonWithRectangle.Content = greenRectangle;
16.     // Button with a Container, StackPanel
17.     Button buttonWithStackPanel = new Button();
18.
    buttonWithStackPanel.Margin = new Thickness(10, 10, 350, 310);
19.
    // Create a StackPanel and set its orinetation to horizontal
```

```
20.      StackPanel stackPanel = new StackPanel();
21.      stackPanel.Orientation = Orientation.Horizontal;
22.      // Create an Ellipse
23.      Ellipse redEllipse = new Ellipse();
24.      redEllipse.Width = 60;
25.      redEllipse.Height = 60;
26.      redEllipse.Fill = Brushes.Red;
27.      // Add to StackPanel
28.      stackPanel.Children.Add(redEllipse);
29.      // Create a TextBlock
30.      TextBlock textBlock1 = new TextBlock();
31.      textBlock1.TextAlignment = TextAlignment.Left;
32.      textBlock1.Text = "Red Circle";
33.      // Add to StackPanel
34.      stackPanel.Children.Add(textBlock1);
35.      // Create a TextBlock
36.      TextBlock space = new TextBlock();
37.      space.TextAlignment = TextAlignment.Center;
38.      space.Text = " ";
39.      // Add to StackPanel
40.      stackPanel.Children.Add(space);
41.      // Create a Rectangle
42.      Rectangle greenRectangle2 = new Rectangle();
43.      greenRectangle2.Height = 60;
44.      greenRectangle2.Width = 120;
45.      greenRectangle2.Fill = Brushes.Green;
46.      // Add to StackPanel
47.      stackPanel.Children.Add(greenRectangle2);
48.      // Create a TextBlock
49.      TextBlock textBlock2 = new TextBlock();
50.      textBlock2.TextAlignment = TextAlignment.Left;
51.      textBlock2.Text = "Green Rectangle";
52.      // Add to StackPanel
53.      stackPanel.Children.Add(textBlock2);
54.      // Set StackPaenl as Button.Content
55.      buttonWithStackPanel.Content = stackPanel;
56.      // Add dynamic button controls to the Window
57.      ParentGrid.Children.Add(helloButton);
58.      ParentGrid.Children.Add(buttonWithRectangle);
59.      ParentGrid.Children.Add(buttonWithStackPanel);
```

Event XAML

Kontrol Windows memiliki sebagian besar event umum seperti Klik, GotFocus, LostFocus, KeyUp, KeyDown, MouseEnter, MouseLeave, MouseLeftButtonDown,

MouseRightButtonDown dan MouseMove. Suatu event di XAML memiliki event yang didefinisikan dalam code(C#) dan kode dieksekusi ketika event dijalankan.

Contoh.

Buat Aplikasi WPF baru dan tambahkan Tombol dan kontrol TextBlock ke Jendela. Posisikan dan format kontrol sesuai keinginan Anda. Kode terakhir saya tercantum di bawah ini.

```
<Grid >
    <Button x:Name="HelloButton" Content="Click Me"
       Width="150" Height="40" Margin="11,10,357.667,280.667"
       FontSize="16" />
    <TextBlock x:Name="HelloTextBlock" Width="400" Height="100"
       Margin="10,57,208.667,163.667" Background="LightGray"
       FontSize="30" Foreground="Orange"/>
</Grid>
```

Kita sekarang akan menambahkan pengendali event Klik tombol dan menulis kode yang akan menambahkan beberapa teks ke TextBlock pada klik tombol. Seperti yang ditunjukkan dalam kode berikut, tambahkan hander klik acara yang disebut HelloButton_Click.

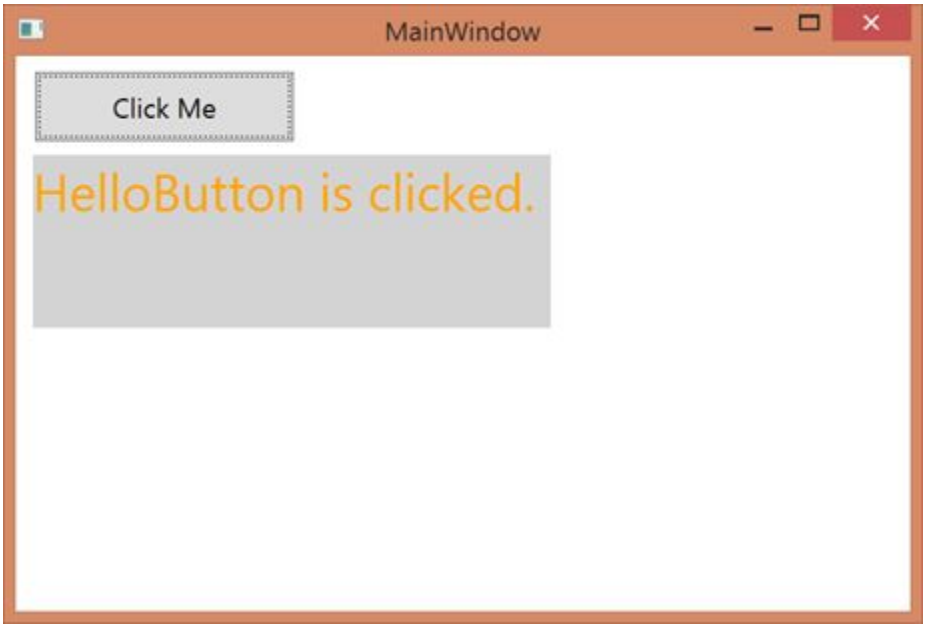
```
<Button x:Name="HelloButton" Content="Click Me"
   Width="150" Height="40" Margin="11,10,357.667,280.667"
   FontSize="16" Click="HelloButton_Click" />
```

Sekarang buka code(C#) dan tambahkan daftar kode berikut. Kode ini adalah pengendali event klik HelloButton dan memperbarui Teks dari TextBlock.

```
void HelloButton_Click(object sender, RoutedEventArgs e)
{
    HelloTextBlock.Text = "HelloButton is clicked.";
}

```

Bangun dan jalankan aplikasi dan klik tombol, output akan berubah menjadi sebagai berikut.


MainWindow
Click Me
HelloButton is clicked.

XAML TextBox

Elemen XAML <TextBox /> mewakili kontrol XAML TextBox. Artikel ini membahas cara membuat TextBox dan mengatur berbagai propertinya seperti warna latar belakang dan latar depan, pengaturan ukuran dan posisi, font, pembungkus, gulir, dan validasi input.

Membuat TextBox

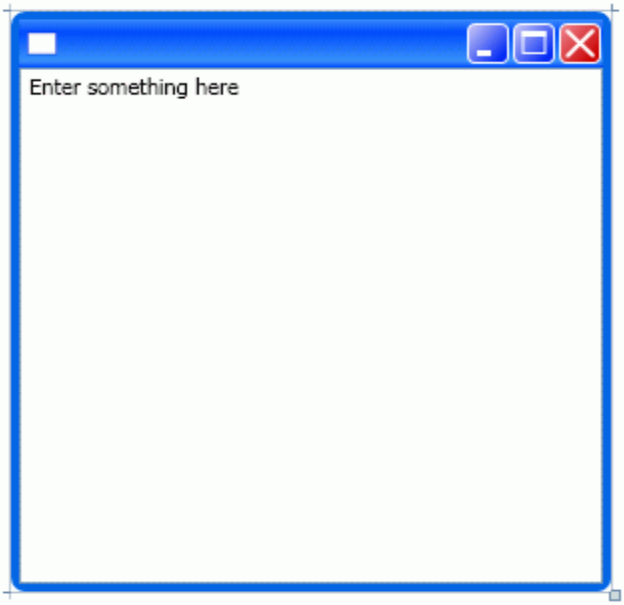
Tag <TextBox /> membuat *Text Box*. Sintaks berikut membuat *Text Box*:

- <TextBox />
- <TextBox> </TextBox>
- <TextBox> Masukkan sesuatu di sini </TextBox>

Potongan kode ini membuat *Text Box* dengan teks di dalamnya.

```
1. <Canvas ID = "root" xmlns
   = "http://schemas.microsoft.com/2003/xaml" xmlns: def
   = "Definition" >
2. <TextBox> Enter something here </TextBox>
3. </Canvas>
```

Output dari potongan kode di atas menghasilkan *Text Box* pada Gambar 1. Posisi default *Text Box* adalah 0,0.



Pengaturan Ukuran dan Posisi

Atribut Height and Width mengatur tinggi dan lebar *Text Box*. Mengatur posisi *Text Box* agak rumit. Untuk mengatur posisi *Text Box*, Anda harus mengatur posisi Top dan Left dari tag XAML *parent*(induk). Misalnya, dalam kasus ini, tag induk untuk TextBox adalah Canvas. Jadi yang diatur atribut Canvas.Top dan Canvas.Left untuk mengatur posisi atas dan kiri *Text Box*.

```
1. <TextBox Width="200" Height="40" Canvas.Top="50" Canvas.Left ="20">
2.   Enter something here
3. </TextBox>
```

Output baru terlihat seperti berikut.

Mengatur Warna Latar dan Latar Depan

Atribut Background dan Foreground mengatur warna latar belakang dan latar depan *Text Box*. Kode berikut menetapkan warna latar belakang sebagai merah dan warna foreground sebagai kuning untuk *Text Box*.

```
1. <TextBox Width="200" Height="40" Canvas.Top="50" Canvas.Left="20"
2.    Background = "Red" Foreground = "Yellow" >
3.    Enter something here
4. </TextBox>
```

Text Box dengan latar depan kuning dan latar belakang merah terlihat seperti gambar berikut ini

Pengaturan Font

Atribut FontFamily, FontSize, FontStyle, dan FontWeight masing-masing mengatur wajah font, ukuran, gaya, dan bobot font *Text Box*.

```
5. <TextBox Width="200" Height="40" Canvas.Top="50" Canvas.Left="20"
6.    Background = "Red" Foreground = "Yellow" >
7.    FontFamily = "Verdana" FontSize = "14" FontStyle
      = "Italic" FontWeight = "ExtraBold" >
8.    Enter something here
9. </TextBox>
```

Ini adalah output dengan font verdana, ukuran 14 poin, style italic, dan bobot ekstra tebal.

Wrapping dan Scroll Teks

Atribut TextWrapping mengatur pembungkus teks dan VerticalScrollBarVisibility dan HorizontalScrollBarVisibility mengatur sroll secara vertikal dan horizontal.

```
1. TextWrapping = "Wrap" VerticalScrollBarVisibility
   = "Auto" HorizontalScrollBarVisibility = "Visible"
```

TextBox baru terlihat seperti Gambar 5.

Membatasi Input Teks

Atribut MaxHeight, MaxWidth, MaxLines, dan MaxLength membatasi tinggi maksimum, lebar maksimum, jumlah maksimum baris, dan panjang maksimum *Text Box*. Demikian pula MinHeight, MinWidht, MinLines, dan MinLength membatasi ketinggian minimum, lebar minimum, jumlah baris minimum, dan panjang minimum *Text Box*.

Mengatur atribut IsReadOnly ke true membuat *Text Box* tidak dapat diedit.

Referensi

https://www.c-sharpcorner.com/blogs/what-is-xaml122

Proses Desain UX (User Experience)

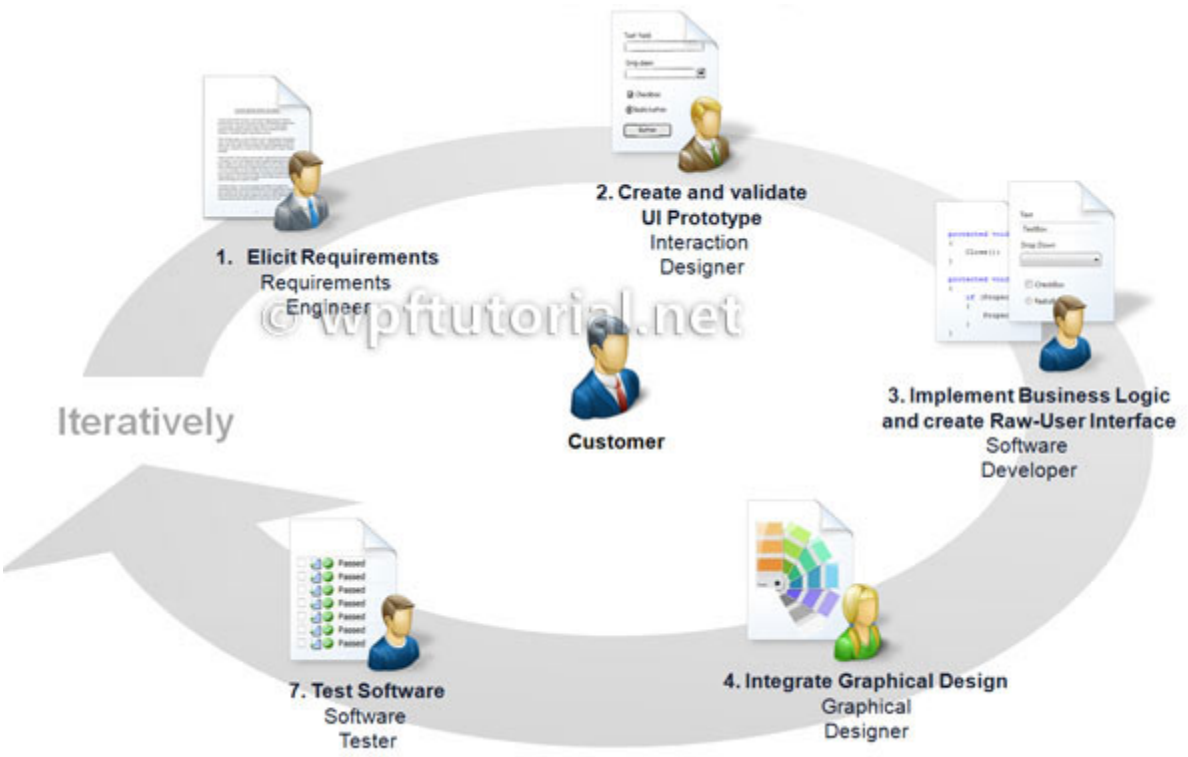
UX menjadi Faktor Sukses Utama

Di masa lalu, fokus utama pengembangan produk adalab membangun produk yang memenuhi persyaratan fungsional pengguna. Pengalaman pengguna sering dianggap terlambat dalam proses pengembangan. Tetapi sekarang ini pelanggan menuntut lebih dari sekedar produk yang berfungsi. Memberikan fitur yang tepat masih merupakan prasyarat untuk produk yang bagus, tetapi untuk mengubahnya menjadi sesuatu yang luar biasa, Anda perlu memberikan UX yang baik!

Memberikan UX yang kaya bukanlah hal yang menguntungkan. Itu perlu direncanakan, dirancang dan diintegrasikan ke dalam pengembangan suatu produk. Merancang pengalaman pengguna yang kaya bukan hanya tentang membuat antarmuka pengguna dengan beberapa grafik dan gradient melainkan konsep yang jauh lebih luas. Memberikan UX yang kaya lebih tentang menciptakan koneksi emosional antara pengguna dan perangkat lunak. Itu membuat pengguna merasa senang dan dia suka terus menggunakan perangkat lunak tersebut.

Alur Kerja Pengembangan Proyek WPF

Mengembangkan aplikasi WPF dengan pengalaman pengguna yang kaya membutuhkan keterampilan lebih dari sekadar analis persyaratan yang menetapkan daftar kasus penggunaan dan pengembang yang mengimplementasikan perangkat lunak. Anda harus mencari tahu apa yang benar-benar dibutuhkan pengguna. Ini dapat dilakukan dengan mengikuti pendekatan yang berpusat pada pengguna.



1. Persyaratan Elicit

Seperti halnya proyek perangkat lunak apa pun, penting untuk mengetahui dan memfokuskan target pengembangan Anda. Anda harus berbicara dengan para pemangku kepentingan dan pengguna untuk mengetahui kebutuhan yang sebenarnya. Kebutuhan ini harus disempurnakan menjadi fitur dan diekspresikan dalam kasus penggunaan (abstrak) atau skenario pengguna (ilustrasi). Prioritaskan tugas-tugas berdasarkan risiko dan kepentingan dan bekerjalah secara iteratif. Pekerjaan ini dilakukan oleh peran requirement engineering.

2. Buat dan Validasi Prototipe UI

Membuat prototipe antarmuka pengguna adalah langkah penting untuk berbagi ide antara pengguna dan engineer untuk menciptakan pemahaman umum tentang desain interaksi. Tugas ini biasanya dilakukan oleh perancang interaksi. Sangat membantu untuk hanya membuat sketsa antarmuka pengguna dengan cara kasar untuk mencegah diskusi awal tentang detail desain. Ada beberapa teknik dan alat untuk melakukan ini. Beberapa dari mereka adalah:

- **Prototipe kertas**

  Gunakan kertas dan pensil untuk menggambar sketsa kasar antarmuka pengguna Anda. Tidak diperlukan alat dan infrastruktur. Semua orang bisa menuliskan ide-ide mereka di atas kertas.
- **Wireframes**

  Wireframes sering digunakan untuk membuat sketsa tata letak halaman. Ini disebut gambar

rangka karena Anda hanya menggambar garis besar kontrol dan gambar. Ini dapat dilakukan dengan alat-alat seperti PowerPoint atau Visio

- **Expression Blend 3 - Sketch Flow** Sketch flow adalah fitur keren baru untuk membuat prototipe interaktif langsung di WPF. Anda dapat menggunakan "gaya wiggly" terpadu untuk membuatnya terlihat samar. Prototipe dapat dijalankan dalam pemain mandiri yang memiliki mekanisme umpan balik terintegrasi.
- **Prototipe Interaktif** Pendekatan yang paling mahal dan nyata adalah membuat prototipe interaktif (dapat digunakan kembali) yang berfungsi sebagai aplikasi nyata tetapi dengan data tiruan.

Sangat disarankan untuk menguji prototipe UI Anda pada pengguna sungguhan. Ini membantu Anda untuk mencari tahu dan mengatasi masalah desain di awal proses pengembangan. Teknik-teknik berikut ini sangat populer untuk mengevaluasi prototipe UI:

- **Walktrough**
  Walktrough biasanya dilakukan pada awal proyek dengan wireframes atau prototipe kertas. Pengguna mendapat tugas untuk dipecahkan dan ia mengendalikan prototipe dengan menyentuh di atas kertas. Pemimpin tes daripada menyajikan makalah baru yang menunjukkan keadaan setelah interaksi.
- **Lab Kegunaan**
  Untuk melakukan lab kegunaan, Anda memerlukan komputer dengan perangkat lunak tangkapan layar dan kamera. Proband mendapat tugas yang harus dilakukan dan persyaratan serta insinyur interaksi mengawasinya melakukan hal ini. Mereka seharusnya tidak berbicara dengannya untuk mencari tahu di mana dia terjebak dan mengapa.

3. Menerapkan Logika Bisnis dan Antarmuka Pengguna Mentah

4. Mengintegrasikan Desain Grafis

5. Uji perangkat lunak

Peran

Buliding antarmuka pengguna modern dengan pengalaman pengguna yang kaya membutuhkan keterampilan tambahan dari tim pengembangan Anda. Keterampilan ini digambarkan sebagai peran yang dapat didistribusikan di antara orang-orang di tim pengembangan Anda.

- **Pengembang**
  Pengembang bertanggung jawab untuk mengimplementasikan fungsionalitas aplikasi. Dia

menciptakan model data, mengimplementasikan logika bisnis dan semua kabel ke tampilan yang sederhana.

- **Desainer Grafis**
  Perancang grafis bertanggung jawab untuk membuat konsep grafis dan membangun aset grafis seperti ikon, logo, model 3D, atau skema warna. Jika desainer grafis terbiasa dengan alat Microsoft Expression, ia langsung membuat gaya dan template kontrol.
- **Desainer Interaksi**
  Perancang interaksi bertanggung jawab atas konten dan aliran antarmuka pengguna. Dia membuat wireframes atau sketsa UI untuk berbagi idenya dengan tim atau pelanggan. Dia harus memvalidasi karyanya dengan melakukan walktroughs atau storyboard.
- **Integrator**
  Integrator adalah artis antara desainer dan dunia pengembang. Dia mengambil aset dari desainer grafis dan mengintegrasikannya ke dalam antarmuka pengguna mentah pengembang. Peran ini membutuhkan seperangkat keterampilan yang langka sehingga seringkali sulit untuk menemukan orang yang tepat untuk itu.