آموزش WPF :

یکی از چیزهایی که هنگام کار باwpf با آن مواجه خواهیم شد xaml است. Xaml مخفف Extensible Application Markup Language است. این یک زبان ساده و بیانی بر اساس xml است.

در xaml ایجاد، مقدار دهی اولیه و تنظیم خصوصیات اشیاء با روابط سلسله مراتبی بسیار آسان است. عمدتا برای طراحی رابط کاربری گرافیکی استفاده می شود.

وقتی که پروژه wpf ایجاد میکنیم برخی از کدهای xaml به صورت پیش فرص ساخته میشوند که شرح هر دستور به صورت زیر است :

Window : این دستور برای باز کردن یک شی از جنس wpf است.

x:Class = “Resources Main Window” : این کد در واقع رابط بین کلاس و پنچره ای است که اشیا رو داخلش قرار میدیم. اینکه کدهایی که مینویسیم بفهمه روی کدوم پنچره اجرا بشه از اینجا مشخص میشه.

Xmlns : در واقع برای اینکه بتونیم از زبان xaml در برنامه استفاده کنیم باید با این دستور فضای نامش رو اضافه کنیم.

چند دستور دیگر هم با xmlns اضافه شده اند که اینها هم برای اضافه کردن فضای نام استفاده میشوند.

Grid : در واقع همه تگ هایی که قراره در پروژه استفاده کنیم در بین این تگ قرار میگیره.

در زبان Xaml به دو روش میتونیم که المان رو تعریف کنیم، مثلا برای تعریف یک کلید:

<Button content = “pull” Height = “30” Width = “60” />

روش دوم :

<Button>

<Button.Content>push</Button.Content>

<Button.Height>30</Button.Height>

<Button.Width>60<Button.Width>

</Button>

زبان xaml داری چینش درختی هست که در ویژوال استادیو میتونیم این رو به صورت ببینیم:

یکی در کد نویسی xmal میتونیم ببینیم.

و درگیری استفاده از ابزاری هست که ویژوال استادیو در اختیارمون قرار میده به نام Live visual tree.

چرا باید از dependency properties استفاده کنیم :

وقتی میخواهیم استایلی رو به تگی نسبت بدیم باید ازش استفاده کنیم.

وقتی میخواهیم وارد کردن داده ها الزامی باشند، مثلا در ورودی ها.

وقتی میخواهیم از منبع ایستا یا پویا استفاده کنیم.

وقتی میخواهیم از انیمیشن استفاده کنیم.

Dependency Properties قابلیت های زیادی را در اختیار ما قرار می دهند.

**مقدماتی :**

**جلسه اول :**

برای ساخت یک اپلیکیشن Wpf با ویژوال استادیو باید یک پروژه از نوع WPF Application بسازیم. وقتی پروژه wpf ساخته شد یک فرم میبینیم که این فرم یک از یک صفحه فرم که روش یه grid هست تشکیل شده.

پایین صفحه یک گزینه zoom هست که میتونیم اندازه بخش گرافیکی رو کوچیک یا بزرگ کنیم. هر فرم یک سری properties داره که در زیر گزینه هایش رو بررسی میکنیم :

Icon : برای گذاشتن آیکون استفاده میشه.

ShowInTaskbar : برای نمایش یا عدم نمایش تسک بار استفاده میشه.

SizeToContent: سایر فرم رو میتونیم مشخص کنیم.

Title : عنوان فرم هست.

WindowStart : میتونیم مشخص کنیم که موقعی که پروژه اجرا میشه فرم در کجای صفحه نمایش دیده بشه.

WindowState : حالت فرم رو مشخص میکنه، که میتونیم مشخص کنیم که فرم تمام صفحه باشه یا اندازه ای که مشخص کردیم یا به نحوه های دیگر.

Toltip : هم چیزی رو که داخلش مینویسیم وقتی با موس روی فرم میریم برامون نمایش میده.

Brush : برای رنگ کردن فرم استفاده میشه.

AllowsTransphrem : برای برداشتن گزینه های minimaiz، maximaiz و close در بالای فرم استفاده میشه.

ویژگی های button :

Opacity : میزان شفافیت دکمه رو میتونیم مشخص کنیم.

روی هر المانی میتونیم کلیک راست کنیم و گزینه view code رو بزنیم و به بخش کدهای اون دسترسی داشته باشیم یا با دبل کلیک کردن روی هر المان به متد مربوط به اون دسترسی داشته باشیم.

روی button دبل کلیک میکنیم تا به متد مربوط به اون برسیم و حالا دستوراتی رو که میخوایم هنگام کلیک کردن روی این button اجرا بشن رو داخل متد مینویسیم.

مثال:

private void btn1\_Click(object sender, RoutedEventArgs e)

{

MessageBox.Show(“ به سایت خوش آمدید!”);

}

حالا با کلیک کردن روی کلید یک پنچره باز میشود و متن بالا رو نمایش میده.

**کنترل Label :** برای نمایش یک برچسب استفاده میشه و خواص رنگ و سایز و ... رو داره. مثلا برای راهنمای نرم افزار میشه ازش استفاده کرد.

همچنین میتونیم فونت هم براش انتخاب کنیم.

نکته : میتونیم از label به عنوان دکمه هم استفاده کنیم. برای این کار باید در بخش properties قسمت Eveantها، خاصیت مربوط به دبل کلیک رو پیدا و روش کلیک میکنیم و متد مربوط به کلیک برای ما ایجاد میشه و دستورات خودمون رو داخلش مینویسیم.

**کنترل TextBox :** وقتی که این کنترل رو اضافه میکنیم به فرم یه متن پیش فرض داخلش هست برای پاک کردن این متن در قسمت properties متن داخل منوی Text رو پاک میکنیم.

میتونیم فونت براش در نظر بگیرم.

خاصیت ToolTip داره. رنگ پس زمینه، رنگ متن و ... اندازه فونت هم میتونیم مشخص کنیم.

منوی Selection Brush: وقتی که متن نوشته شده داخل textbox رو انتخاب میکنیم حالت پیش فرض آبی هست ولی میتونیم این رنگ رو با این منو به رنگ دلخواه تغییر بدیم.

**کنترل RadioButton :** خاصیت های کنترل های قبل مثل رنگ، نام و ... داره.

منوی IsCheckid : این گزینه رو اگر فعال کنیم وقتی برنامه رو اجرا میکنیم redioButton تیک خورده است.

**کنترل CheckBox :** فرق چک باکس با Radiobutton این هست که در چک باکس میتونیم چند تاشون رو تیک بزنیم.

**کنترل GroupBox :** برای دسته بندی کنترل ها به کار میره.

**کنترل Password Box :** یک نوع TextBox هست با این تفاوت که وقتی داخلش بنویسیم متن نمایش داده نمیشه، برای اینکه مشخص کنیم حروف رو به چه صورت نمایش بده مثلا ستاره یا چیزهای دیگه در Properties در منوی Password این رو مشخص میکنیم.

**کنترل Image :** که برای نمایش تصویر استفاده میشه. از قسمت Properties و منوی Source میتونیم آدرس تصویر رو مشخص کنیم.

منوی Sretchکه میتونیم اندازه تصویر رو مشخص کنیم که چقدر از کنترل image رو پوشش بده.

**کنترل Date Picker :**  که تاریخ و ساعت رو میتونیم در این کنترل وارد کنیم.

**کنترل Progress Bar :** برای این هست که مثلا وقتی که منتظر لود یک اطلاعاتی هستیم یک کشو میاد و کم کم پر میشه. مثل وقتی داریم چیزی رو کپی میکنیم.

مثال : میخواهیم یک دکمه داشته باشیم با یک پروکرس بار که زمانی که روی دکمه کلیک کردیم به مقدار پروکرس بار 5 تا اضافه بشه.

private void btn1\_Click(object sender, RoutedEventArgs e)

{

prc1.Value += 5;

}

Prc1 در واقع نامی هست که برای ProgressBar در نظر گرفته ایم.

**کنترل DockPanel:**

<dockPanel>

<Menu x:Name=”doc1” DockPanel.Dock=”Top”>

<MenuItem Header=” رنگ”>

<MenuItem Header = “انتخاب”>

<MenuItem.Icon>

<Image Source =”آدرس عکس با پسوند”></Image>

</MenuItem.Icon>

<MenuItem Header=”آبی” Click=”itm1\_Click” InputGestureText=”ctrl+B”>

<MenuItem.Icon> <Image Source = “…” ></Image> </MenuItem.Icon>

</MenuItem>

<MenuItem Header=”قرمز” Click=”itm2\_Click” InputGestureText=”ctrl+R”>

<MenuItem.Icon> <Image Source = “…” ></Image> </MenuItem.Icon>

</MenuItem>

<MenuItem Header=”زرد” Click=”itm3\_Click” InputGestureText=”ctrl+y”>

<MenuItem.Icon> <Image Source = “…” ></Image> </MenuItem.Icon>

</MenuItem>

</MenuItem>

<MenuItem Header= “خروج” Click=”itm4\_Click”>

<MenuItem.Icon>

<Image Source=”…”></Image>

</MenuItem.Icon>

</MenuItem>

</MenuItem>

</Menu>

</dockPanel>

قبل از اجرای برنامه باید برای رویدادهای کلیک متد بسازیم تا برنامه با خطا مواجه نشود.

حالا متد های هر کلید :

private void itm1\_Click(object sender, RoutedEventArgs e) {

doc1.Background = Brushes.Blue; }

private void itm2\_Click(object sender, RoutedEventArgs e) {

doc1.Background = Brushes.Red; }

private void itm3\_Click(object sender, RoutedEventArgs e) {

doc1.Background = Brushes.Yellow; }

private void itm4\_Click(object sender, RoutedEventArgs e) {

this.Close; }

**کنترل TabControl :**

این کنترل پیش فرض دوتا تب داره. هر تب یک Grid هم داره برای ایجاد تب های دیگه.

برای اینکه تمایز قایل بشیم بین تب ها بکگراند و نامشون رو تغییر میدیم. حالا در یکی از تب ها به صورت زیر تصویری رو قرار میدیم :

<Grid>

<TabControl x:Name=”tabControl” HorizontalAlignment=”Left” Height=”361” …>

<TabItem x:Name=”tabl1” Header=”TabItem”>

<Gird>

<Grid.Background>

<ImageBrush ImageSource=” image url” > </ImageBrush>

</Grid.Background>

</Grid>

</Grid>

میخوایم یک کلید در تب اول درست کنیم که با کلیک روی اون به تب دوم بریم:

private void button1\_Click(object sender, RoutedEventArgs e) {

tabControl.SelectedItem = tab2; }

tabl2 در واقع نام تب دم هست.

**به دست آوردن تاریخ فارسی :**

اول یک کنترل textBox و یک button به پروژه اضافه میکنیم. نام button رو به نمایش تاریخ روز تغییر میدیم.

حالا کدی که باید داخل متد کلید بنویسیم :

private void button1\_Click(object sender, RoutedEventArgs e) {

System.Globalization.PersianCalendar p = new System.Globalization.PersionCalendar();

textBox1.Text = p.GetYear(DateTime.Now).ToString() + p.GetMonth(DateTime.Now).ToString(00) +

p.GetDayOfMonth(DateTime.Now).ToString(00);

}

حال فرضا میخوایم داخل یک متد یک فرم از یک پروژه دیگه رو باز کنیم :

private void button3\_Click(object sender, RoutedEventArgs e) {

Window1 win = new Window1();

Win.textBox1.Text = textBox1.Text;

Win.ShowDialog(); }

**جلسه سوم :**

نکته : زمانی که متنی رو داخل Label قرار میدیم این متن وسط کنترل label قرار نمیگیره، برای اینکه متن وسط لیبل قرار بگیره اتریبیوت زیر رو به تگش اضافه میکنیم :

HorizontalContentAlignment=”Center”

**ایجاد تایمر با کنترل :**

برای درست کردن یک تایمر از دستور زیر استفاده میکنیم :

System.Windows.Threading.DispatcherTimer timer2 = new System.Windows.Threading.DispatcherTimer();

حالا رویداد تیک رو برای تایمر باید بنویسیم :

void timer2\_Tick(object sender, EventArgs e) {

pc.Value +=10;

if (pc.Value == 100) {

timer2.Stop();

new MainWindow().ShowDialog();

this.Close();

}

}

نکته : اگر در پروژه ای چند پنچره مختلف داشته باشیم و بخواهیم زمان اجرای پروژه پنچره خاصی رو ابتدا باز کنه در فایل App.xaml در قسمت StartupUri نام پنچره مورد نظر رو وارد میکنیم.

**رسم اشکال هندسی :**

اشکال هندسی رو داخل تگ StackPanel رسم میکنیم.

برای رسم بیضی و دایره از تگ Ellipse استفاده میکنیم:

<StackPanel>

<Ellipse margin =30 Fill=”Blue” Height=”50” Width=”150” StrokeThickness=”3” Stroke=”Red” />

</StackPanel>

StrokThickness ضخامت خطی هست که دور دایره یا بیضی میفته. و Stroke رنگ خط دور دایره رو مشخص میکنه.

**Observable collection :** در pwf نوعی لیست ویژه هست که زمانیکه به کنترلی بایند شد، کنترل را از تغییرات خودش آگاه میکند. برای مثال اگر آیتمی به این لیست اضافه شد بلافاصله آن آیتم را در کنترل مقید به آن نیز خواهید دید. به همین ترتیب در مورد ویرایش و حذف بدون کوچکترین تماسی با کنترل مورد نظر، نتیجه نهایی رو در کنترل خواهیم دید.

**Dependency Property :** نوع خاصی از پراپرتی هست که توسعه یافته CLR Property است ( CLR Property رو در بخش بعد توضیح میدهیم).

کلاسی که بخواهد در داخل خود Dependency Property تعریف کند باید از کلاس Dependency Object ارث بری کند. اکثر کلاس های تعریف شده در wpf که مربوط به UI هستند، از این کلاس ارث بری کرده اندو به طور پیش فرض از این نوع خاصیت ها پشتیبانی می کنند.

برای مثال IsMouseOver یک Dependency Property است که در اکثر کنترل ها وجود دارد.

x:Type : این یک Markup Extension است که عملکرد مشابه typeof در سی شارپ را دارد و زمانیکه از آن استفاده کنیم که بخواهیم نوع کنترل را مشخص کنیم.

کاربرد dependency Property ها :

1. زمان تنظیم استایل برای یک کنترل
2. زمان بایند کردن داده ها
3. زمان تنظیم منابع به صورت Static یا dynamic
4. زمانیکه بخواهیم از animation پشتیبانی کنیم

برای مثال میخواهیم یک خاصیت به نام Text و از نوع dependency property را به کنترل SampleControl اضافه کنیم:

public static readonly DependencyProperty TextProperty =

DependencyProperty.Register(nameof(Text) ,

typeof(string), typeof(SampleControl),

new UIPropertyMetadata(default));

سرفصل های دوره حکیم :

1. WPF چیست؟
2. Button ها
3. طراحی رابط کاربری
4. کنترل Grid
5. آموزش صفحه آرایی با Grid
6. تابع DragMove برای حرکت دادن پنچره
7. آموزش کنترل StackPanel
8. آموزش Slider Control و ساخت پروژه Color Generator
9. پروژه ساخت نمودار میله ای
10. کنترل Datagrid
11. Storyboard Animations
12. DataContext در wpf
13. TextBoxو RichTextBox
14. Data Trigger
15. نمایش تاریخ و زمان با فرمت دلخواه
16. Data Binding
17. ساخت Toast notifications اندروید
18. آموزش کنترل Border
19. شخصی سازی button
20. کنترل TreeView
21. Format String
22. Property Trigger
23. آموزش متصل کردن دو کنترل
24. آموزش قرار دادن لینک
25. کنترل Calendar
26. ساخت Config File
27. مبدل مقادیر IvalueConverter
28. استفاده از Styles سبک ها
29. کنترل Window
30. کنترل Radio Button و CheckedBox
31. پردازش متن Text Rendering
32. خاصیت Name
33. کار با پس زمینه
34. بررسی Events
35. کار با فایل XML
36. آموزش Stack Panel
37. آموزش ListBox
38. آموزش کامل WrapPanel
39. کنترل ToolTip
40. آموزش DockPanel
41. کنترل کننده رویداد
42. کنترل StackPanel
43. کنترل WarpPanel
44. مدیریت طرح بندی و چیدمان عناصر
45. آموزش ساخت ScrollBar
46. کلاس MessageBox
47. معرفی خواص کنترل ها
48. کنترل Canvas
49. نامگذاری کنترل ها
50. کنترل TextBlock
51. آموزش Canvas
52. کنترل DockPanel
53. آموزش GroupBox
54. کنترل Expander
55. کنترل Grid
56. Label و کلیدهای دسترسی سریع
57. محرک های مبتنی بر رویداد
58. آموزش کار با Xpathدر XML
59. قرار دادن کنترل های دیگر در Label
60. طراحی قالب button به وسیله Style
61. تغییر قلم متن یک کنترل
62. آموزش تگ های Spanو Run در TextBlock
63. تراز بندی متن در Text Block
64. دستور boldو italic و underline و تگ های تو در تو

**جلسه اول : wpf چیست؟**

User Interface: به معنی رابط کاربری هست. که همون زبان سمت فرانت اند ما هست.

هر پروژه wpf دارای یک قسمت کد Xaml است که با کمک اون میتونیم پنچره مون رو طراحی کنیم. هر پروژه pwf دارای یک پنچره هست که این پنچره خواصی رو داره، مثل Title، width ، width، WindowStartupLocalion، windowState و ...

پروپرتی های مختلفی داره که در قسمت Properties میتونیم ویژگی های مربوط به هر تگ رو ببینیم.

پنچره گرافیکی رو میتونیم با دستور Gird بخش بخندی کنیم برای اینکار از دستور زیر استفاده میکنیم :

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width=”1\*” />

<ColumnDefinition Width=”3\*” />

</Grid.ColumnDefinitions>

</Grid>

با دستورات بالا صفحه رو به دو ستون تقسیم بندی کردیم که ستون دوم همیشه سه برابر نسبت به ستون اول صفحه رو اشغال میکنه، همچنین میتونیم برای اندازه دهی به هر بخش از واحد پیکسل هم استفاده کنیم.

برای اینکه بخایم یک کنترل رو داخل یک ستون خاص قرار بدیم به صورت زیر عمل میکنیم :

<Button Grid.Column=”0” Width=”150” Height=”50”>

Click Mi!

</Button>

**جلسه دوم : Button ها**

در Xaml دستورات رو داخل تگ قرار میدیم. که هر تگ میتونه اتریبیوت های خاص خودش رو داشته باشه.

به کلید هامون میتونیم رویدادهایی رو وصل کنیم مثل رویداد کلیک:

<Button Click = “ “> Click Mi! </Button>

حالا داخل click نام متدی که میخایم وقتی کاربر روی کلید کلیک کرد دستورات اون متد اجرا بشه رو مینویسیم. حالا تابعی که میسازه:

private void Button\_Click\_1(object sender, RoutedEventArgs e) {

MessageBox.Show(“hello hashem”); }

اتریبیوت Content : به وسیله این اتریبیوت میتونیم نام کلید رو مشخص کنیم.

اتریبیوت FontSize : برای اینکه مشخص کنیم سایز متن کلید چقدر باشه.

Background : برای مشخص کردن رنگ بکگراند کلید استفاده میشه.

Foreground : برای مشخص کردن رنگ متن کلید استفاده میشه.

FontWeight : برای بلد کردن، لایت کردن یا موارد دیگر، متن کلید استفاده میشه.

نکته : اگر از Content استفاده میکنیم نباید متن رو بین دو تک button دوباره بنوسیم.

همچنین میتونیم به جای متن از تصویر استفاده کنیم.

همچنین میتونیم موقعیت های المنت های داخل کلید رو با استفاده از Padding تغییر دهیم.

برای اینکه گوشه های کلید رو به صورت منحنی درست کنیم به صورت زیر عمل میکنیم :

<Border Width=”250” Height=”65” Background = “Block” BorderThickness = “5” CornerRadius = “25”>

<Button Padding=”0,0,55,0” FontWeight= “UltralLIght” Height =”50” FontSize=”18” Click=”Button\_Click\_1”>

Click Mi!

</Button>

</Border>

Cursor : زمانیکه با موس روی کلید بریم شکل موس به شکل دست در میاد.

برای ایجاد Radius روی کلید بهتر است از روش زیر استفاده کنیم :

<Button Content=”Click Mi”>

<Button.Resources>

<Syle TargetType=”Border”>

<Setter Property=”CornerRadius” Value=”10”></Setter>

</Style>

</Button.Resources>

</Button>

**جلسه سوم : طراحی رابط کاربری با سی شارپ**

اگر در TextBlock از اتریبیوت TextWrapping =”Wrap” استفاده کنیم باعث میشه زمانی که نوشته به اخر خط برسه خودش به خط بعد بره.

اگر در TextBlock بخوایم در یک قسمت از نوشته استایل خاصی داشته باشیم اون قسمت رو داخل Run میگذاریم و استایل مورد نظر رو بهش میدیم.

مثلا برای اینکه بخایم زیر قسمتی از نوشته خط بکشیم از اتریبیوت TextDecorations =”underline” استفاده میکنیم.

میتونیم دستورات xmal رو به معادل سی شارپ هم تبدیل کنیم، مثال :

public partial class MainWindow : window {

public MainWindow () {

InitializeComponent ();

TextBlock textBlock = new TextBlock ();

textBlock.Textwrapping = TextWrapping.wrap;

textBlock.Margin = new Thickness (10);

…

**جلسه چهار : کنترل گرید**

با گرید میتونیم صفحه خودمون رو به سطر و ستون هایی تقسیم بندی کنیم برای اینکار از روش زیر استفاده میکنیم :

<Grid>

<Grid.Column