# wpf

## 工程转换

### 属性

控制台程序属性输出类型可以选择Windows应用程序，就会去掉控制台

依赖的dll：

PersentationCore WPF的核心类库

PersentationFramework封装了与WPF控件相关类型的类库

System.Xaml XAML解析的类库

WindowBase Window窗体相关的类库

### 简单创建

[STAThread]

static void Main(string[] args)

{

Window mainWindow = new Window();

mainWindow.Title = "WPF应用";

//Application类型用于创建一个消息循环

Application app = new Application();

app.Run(mainWindow);

}

### APP使用

[STAThread]

static void Main(string[] args)

{

App app = new App();

app.Run();

}

public class App : Application

{

protected override void OnStartup(StartupEventArgs e)

{

base.OnStartup(e);

MainWindow mainWindow = new MainWindow(); //窗口可以创建类，也可以用创建的wpf窗体（控制台不能创建窗体可以创建UserControl1进行改造）

mainWindow.Show();

}

}

public class MainWindow : Window

{

public MainWindow()

{

this.Title = "MainWindow";

this.MouseLeftButtonDown += new System.Windows.Input.MouseButtonEventHandler(MainWindow\_MouseLeftButtonDown);

}

void MainWindow\_MouseLeftButtonDown(object sender, System.Windows.Input.MouseButtonEventArgs e)

{

MessageBox.Show(e.GetPosition(this).ToString(), this.Title);

}

}

## 新建窗口

添加窗口(WPF)，Window1.xaml

Window1 w = new Window1 ();

w.Show(); //非模式对话框，需要在ui线程创建

if (true != ff.ShowDialog()) return; //模式对话框，自带ui线程

this.DialogResult = true;

## UserControl

### 用户控件创建

新建UserControl1（命名空间UserSpace）

代码中包含：

public int userVal { get; set; }

public void Fun()

{

int y = userVal + 10;

}

### 调用

<Window .........

xmlns:NNN="clr-namespace:UserSpace"

..........>

<NNN:UserControl1 x:Name="userCtl" userVal="123"/>

调用用户控件中的方法

userCtl.Fun();

### 容器

<ContentPresenter Content="{Binding CONTENT\_PAGE}"/>

UserControl \_content;

public UserControl CONTENT\_PAGE

{

get { return \_content; }

set

{

\_content = value;

RaisePropertyChanged("CONTENT\_PAGE");

}

}

CONTENT\_PAGE = userCtl;

## 显示

### 全屏

WindowStyle="None" WindowState="Maximized"

### 去除边框

WindowStyle="None"

AllowsTransparency="True"

### 鼠标拖拽

MouseDown="Window\_MouseDown" this.DragMove();

### 透明背景

WindowStyle="None"

Background="Transparent"

AllowsTransparency="True"

Opacity="1"

### 背景颜色

Background = Brushes.Green;

### 窗口居中

WindowStartupLocation = WindowStartupLocation.CenterScreen;

### 顶层窗口

Topmost="True"

### 屏幕大小

double height = SystemParameters.PrimaryScreenHeight;

double width = SystemParameters.PrimaryScreenWidth;

### 光标

Cursor="Hand"

Cursor cur = new Cursor("1.cur");

this.Cursor = cur;

### 程序单例

public App()

{

this.Startup += App\_Startup;

}

void App\_Startup(object sender, StartupEventArgs e)

{

bool ret = false;

System.Threading.Mutex mutex = new System.Threading.Mutex(true, "xxxx", out ret);

if(!ret)

{

MessageBox.Show("已经运行");

Environment.Exit(0);

}

}

## 控件

### 线程访问控件

============方法1

delegate void FunCallBack(string s,int num);

void Fun(string s1, string s2)

{ textBox1.Text= s1+ s2;}

线程中调用：

Fun ff = new Fun(fun);

textBox1.Dispatcher.Invoke(ff, new object[] { "aa", "bb" });

============方法2

.net4.5以上

tx.Dispatcher.Invoke(()=> {

tx.Text = "xxx";

});

### 排版

#### Grid

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="3\*"/>

<RowDefinition Height="2\*"/>

</Grid.RowDefinitions>

<Grid Grid.Row="0">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="200" />

<ColumnDefinition />

<ColumnDefinition Width="200" />

</Grid.ColumnDefinitions>

<Button Content="btn1" Grid.Column="0" HorizontalAlignment="Left" Margin="48,60,0,0" VerticalAlignment="Top" Width="75"/>

<Button Content="btn2" Grid.Column="2" HorizontalAlignment="Left" Margin="60,60,0,0" VerticalAlignment="Top" Width="75"/>

</Grid>

<Grid Grid.Row="1">

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition />

<ColumnDefinition />

<ColumnDefinition />

</Grid.ColumnDefinitions>

<Border Grid.Row="0" Grid.Column="1">

<Button Content="btn3" />

</Border>

</Grid>

</Grid>

#### StackPanel

默认垂直排列

<StackPanel Grid.Row="1" Orientation="Horizontal">

<Button Content="btn1" Margin="5" HorizontalAlignment="Left" FontSize="20"/>

<Button Content="btn2" Margin="5" HorizontalAlignment="Left" FontSize="20"/>

</StackPanel>

#### Expander

<Expander IsExpanded="False" Header="标志" BorderBrush="{x:Null}">

。。。。。

</Expander>

#### StatusBar

<StatusBar HorizontalAlignment="Stretch" >

<StatusBarItem>

<TextBlock Text="aaa" />

</StatusBarItem>

<StatusBarItem HorizontalAlignment="Right">

<TextBlock Text="bbb"/>

</StatusBarItem>

</StatusBar>

#### 切分窗口

<Grid >

<Grid.RowDefinitions>

<RowDefinition />

<RowDefinition Height="Auto"/>

<RowDefinition Height="200" MaxHeight="400" MinHeight="100"/>

</Grid.RowDefinitions>

<Grid Grid.Row="0" ></Grid>

<GridSplitter Grid.Row="1" ResizeDirection="Rows" Height="3" HorizontalAlignment="Stretch"/>

<Grid Grid.Row="2"></Grid>

</Grid>

可以不定义GridSplitter所在的Row，直接写在两个Grid 中间

#### grid位置改变

\*\*\*.SetValue(Grid.ColumnProperty, 2);

### 属性关联

<Image Grid.Row="0" Name="img"></Image>

<TextBox Grid.Row="1" Text="aaa" Width="{Binding ElementName=img, Path=ActualWidth}"/>

将TextBox的宽度与Image实际显示的图像宽度进行绑定，宽度会随Image显示图像宽度进行变化

### 事件

#### 双击

private void Img\_MouseDown(object sender, MouseButtonEventArgs e)

{

e.ClickCount >=2

}

#### 位置获取

private void Img\_MouseDown(object sender, MouseButtonEventArgs e)

{

Point pt = e.GetPosition(img);

}

获取鼠标点击位置在img控件中相对位置（GetPosition可以获取相对于某控件的位置）

#### 鼠标滚轮

private void UserControl\_MouseWheel(object sender, System.Windows.Input.MouseWheelEventArgs e)

{

if(e.Delta > 0) {//向上}

else{}

}

### 属性绑定

<TextBox Grid.Row="1" Text="aaa" Width="{Binding WIDTH, Mode=TwoWay}" Name="tb" />

class ViewModel : INotifyPropertyChanged

{

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void OnPropertyChanged(string propertyName = null)

{

if (PropertyChanged != null)

PropertyChanged.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

private int width;

public int WIDTH {

get { return width; }

set

{

width = value;

OnPropertyChanged("Width"); }

}

}

ViewModel viewModel = new ViewModel { WIDTH = 50};

tb.DataContext = viewModel;

设置属性：viewModel.WIDTH = 100;

### Button

<Button Content="btnTxt" Name="btn" HorizontalAlignment="Left" Margin="208,207,0,0" VerticalAlignment="Top" Width="75"/>

### CheckBox

<CheckBox Content="CheckBox" Name="cb" IsChecked="True" HorizontalAlignment="Left" Margin="210,189,0,0" VerticalAlignment="Top"/>

### ComboBox

<ComboBox Name="cbb" HorizontalAlignment="Left" Margin="270,185,0,0" VerticalAlignment="Top" Width="120"/>

cbb.Items.Insert(0,"aa");

cbb.Items.Insert(1, "bb");

int index = cbb.SelectedIndex;

### DataGrid

#### 数据绑定

========combox相关资源定义

xmlns:core="clr-namespace:System;assembly=mscorlib"

<Window.Resources>

<ObjectDataProvider x:Key="SexEnumKey" MethodName="GetValues" ObjectType="{x:Type core:Enum}">

<ObjectDataProvider.MethodParameters>

<x:Type Type="local:SexEnum"/>

</ObjectDataProvider.MethodParameters>

</ObjectDataProvider>

</Window.Resources>

==========前台界面

<DataGrid Name="DataGrid\_Name" ItemsSource="{Binding}" CanUserAddRows="False" AutoGenerateColumns="False" HeadersVisibility="Column">

<DataGrid.Columns>

<DataGridTextColumn Header="姓名" Binding="{Binding Name}" Width="100"/>

<DataGridComboBoxColumn Header="性别" SelectedItemBinding="{Binding Sex}" ItemsSource="{Binding Source={StaticResource SexEnumKey}}" Width="100"/>

</DataGrid.Columns>

</DataGrid>

==========后台后台数据

public enum SexEnum { Male,FeMale};

public class DataItem

{

public string Name { get; set; }

public SexEnum Sex { get; set; }

}

ObservableCollection<DataItem> m\_data = new ObservableCollection<DataItem>();

DataItem cm = new DataItem();

cm.Name = "AA";

cm.Sex = SexEnum.Male;

m\_data.Add(cm);

DataGrid\_Name.DataContext = m\_data;

#### 设置不可编辑

IsReadOnly = true;

#### 禁用用户排序

CanUserSortColumns="False"

#### 背景色

<DataGrid Background="#363636" Foreground="White">

#### 隐藏表头

HeadersVisibility="Column"

#### 选择选择行样式

<DataGrid.RowStyle >

<Style TargetType="DataGridRow">

<Setter Property="Background" Value="White"/>

<Style.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter Property="Background" Value="LightGray"/>

</Trigger>

<Trigger Property="IsSelected" Value="True">

<Setter Property="Background" Value="#90F670"/>

<Setter Property="Foreground" Value="White"/>

</Trigger>

</Style.Triggers>

</Style>

</DataGrid.RowStyle>

<DataGrid.Resources>

<SolidColorBrush x:Key="{x:Static SystemColors.InactiveSelectionHighlightBrushKey}" Color="DodgerBlue"/>

</DataGrid.Resources>

#### 行跳转

dg.SelectedIndex = 123;

dg.ScrollIntoView(dg.SelectedItem);

#### 双击获取行数据

MouseDoubleClick

DataGrid dg = sender as DataGrid;

int index = dg.SelectedIndex;

dgItem item = dg.SelectedItem as dgItem;

if (null == item) return;

判断点击位置的列名：

if (dg.CurrentColumn.Header.ToString() != "列名") return;

#### 获取多行

List<PrjDat> itemList = new List<PrjDat>();

foreach(PrjDat item in PrjDatSouce.SelectedItems)

{

itemList.Add(item);

}

foreach (PrjDat item in itemList)

{

m\_data.Remove(item);

}

### Image

ActualWidth属性表示显示图像后，控件实际的宽度，五图像时该值为0

img.Source = new BitmapImage(new Uri(@"D:\1.jpg", UriKind.RelativeOrAbsolute));

//从文件的byte[]数据数据加载

MemoryStream ms = new MemoryStream(imgDat);

ImageSourceConverter imgSrcCvt = new ImageSourceConverter();

img.Source = imgSrcCvt.ConvertFrom(ms) as System.Windows.Media.Imaging.BitmapFrame;

### Label

<Label Name="Label1" Content="xxxxx" Width="200" Height="80" FontSize="40" FontFamily="Georgia" FontWeight="Bold"/>

设置文字在控件内居中：

HorizontalContentAlignment="Center"

### ListBox

<ListBox Name="listbox1" ></ListBox>

添加方法1：

listbox1.Items.Add(1);

listbox1.Items.Add(2);

添加方法2：

List<int> ll = new List<int>();

ll.Add(2);

ll.Add(3);

listbox1.ItemsSource = null;

listbox1.ItemsSource = ll;

添加方法3：

<ListBox Name="listbox1" DisplayMemberPath="ss"></ListBox>

struct Inf

{

public string ss { get; set; }

}

List<Inf> ll = new List<Inf>();

Inf inf1 = new Inf();

inf1.ss = "aa";

ll.Add(inf1);

listbox1.ItemsSource = null;

listbox1.ItemsSource = ll;

### RadioButton

命名空间内定义数据：

public enum EmunStr

{

XX,

YY,

}

public class EnumToBooleanConverter : IValueConverter

{

object IValueConverter.Convert(object value, Type targetType, object parameter, CultureInfo culture)

{

return value == null ? false : value.Equals(parameter);

}

object IValueConverter.ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

{

return value != null && value.Equals(true) ? parameter : Binding.DoNothing;

}

}

public class DataModel : INotifyPropertyChanged

{

public event PropertyChangedEventHandler PropertyChanged;

private void OnPropertyChanged(string p\_propertyName)

{

PropertyChanged(this, new PropertyChangedEventArgs(p\_propertyName));

}

private EmunStr \_Dat;

public EmunStr Dat

{

get

{

return \_Dat;

}

set

{

\_Dat = value;

OnPropertyChanged("Dat");

}

}

}

Xaml的Window中引入资源

<Window.Resources>

<local:EnumToBooleanConverter x:Key="EnumToBooleanConverter" />

</Window.Resources>

<RadioButton Content="A" IsChecked="{Binding Path=Dat,Converter={StaticResource EnumToBooleanConverter},ConverterParameter={x:Static local:EmunStr.XX}}"></RadioButton>

<RadioButton Content="B" IsChecked="{Binding Path=Dat,Converter={StaticResource EnumToBooleanConverter},ConverterParameter={x:Static local:EmunStr.YY}}"></RadioButton>

<TextBox Grid.Row="1" Text="{Binding Path=Dat}"></TextBox>

DataModel m\_dataModel;

m\_dataModel = new DataModel();

DataContext = m\_dataModel;//设置数据绑定内容

获取选择数据：

m\_dataModel.SampleEnum.ToString()

### Rectangle

<Rectangle Width="200" Height="100" Fill="Blue" Stroke="Black" StrokeThickness="4" />

### StackPanel

默认竖向排布

<StackPanel Orientation="Horizontal">

</StackPanel>

### TextBlock

<TextBlock Name="Promt\_Str" VerticalAlignment="Center" HorizontalAlignment="Center" Text="未定义" FontSize="30" Foreground="#FFE20404" />

使数据在超出行时自动换行

TextWrapping="Wrap"

### RichTextBox

<RichTextBox Name="rr" VerticalScrollBarVisibility="Auto" HorizontalAlignment="Left" Height="474" Margin="128,299,0,0" VerticalAlignment="Top" Width="543" Grid.ColumnSpan="2"/>

rr.Document.Blocks.Add(new Paragraph(new Run("bbbb") { Foreground = Brushes.Green }));

rr.ScrollToEnd();

rr.Document.Blocks.Clear();

### TabControl

<TabControl Name="Tbctl">

<TabItem Height="30" Width="75" Header="项目1">

<Button Content="aaa"></Button>

</TabItem>

<TabItem Height="30" Width="75" Header="项目2" >

<Button Content="bbb"></Button>

</TabItem>

<TabItem Height="30" Width="75" Header="项目3" >

<Button Content="ccc"></Button>

</TabItem>

</TabControl>

### Slider

sl.Minimum= 0;

sl.Maximum= 255;

ValueChanged

Slider sl = sender as Slider;

int val = (int)sl.Value;

### ProgressBar

progress.Maximum = 100;

progress.Value = 2;

### mediaElement

<MediaElement Name="mediaElement1" Stretch="Fill" LoadedBehavior="Manual" />

删除标签中的margin，可以实现填满整个程序

Stretch="Uniform"可以让mediaElement按比例拉伸

打开文件

this.mediaElement1.Source = new Uri("d:\\1.avi");

//可以用File.Exists("d:\\1.avi");验证地址合法性

播放

this.mediaElement1.Play();

暂停

this.mediaElement1.Pause();

停止事件

属性：MediaEnded="mediaElement1\_MediaEnded"

音量

this.mediaElement1.Volume += 10;

媒体长度

TimeSpan span = this.mediaElement1.NaturalDuration.TimeSpan;

string s = string.Format("{0},{1}", span.Minutes, span.Seconds);

当前位置

TimeSpan span = this.mediaElement1.Position;

快进

mediaElement1.Position = mediaElement1.Position + TimeSpan.FromSeconds(10);

设置播放位置

TimeSpan span = new TimeSpan(0, 0, 3);//设置为第3秒

this.mediaElement1.Position = span;

### webBrowser

打开网页

wb.Navigate(new Uri(address));

wb.NavigateToString("<html><body></body></html > ");

跳转

if (wb.CanGoBack){wb.GoBack();}

if (wb.CanGoForward){wb.GoForward();}

导航到新的页面时：

wb.Navigating

导航之后，在下载web页面之前

wb.Navigated

web页面下载完成时

wb.LoadCompleted

### InkCanvas

<Image Name="img" HorizontalAlignment="Center" Stretch="Uniform"/>

<InkCanvas Name="inkCanvas" EditingMode="None" Background="Transparent" Strokes="{Binding InkStrokes, Mode=TwoWay}" HorizontalAlignment="Center"

Width="{Binding ElementName=img, Path=ActualWidth}" Height="{Binding ElementName=img, Path=ActualHeight}" ></InkCanvas>

属性设置：

DrawingAttributes drawingAttributes = new DrawingAttributes

{

Color = Colors.Red,

Width = 2,

Height = 2,

StylusTip = StylusTip.Rectangle,

IsHighlighter = false,

IgnorePressure = true,

};

inkCanvas.DefaultDrawingAttributes = drawingAttributes;

List<System.Windows.Point> pointList = new List<System.Windows.Point>();

绘制椭圆：

double a = 0.5 \* (pt2.X - pt1.X);

double b = 0.5 \* (pt2.Y - pt1.Y);

for (double r = 0; r <= 2 \* Math.PI; r = r + 0.01)

{

pointList.Add(new System.Windows.Point(0.5 \* (pt1.X + pt2.X) + a \* Math.Cos(r), 0.5 \* (pt1.Y + pt2.Y) + b \* Math.Sin(r)));

}

绘制矩形：

pointList.Add(pt1);

pointList.Add(new System.Windows.Point(pt1.X, pt2.Y));

pointList.Add(pt2);

pointList.Add(new System.Windows.Point(pt2.X, pt1.Y));

pointList.Add(pt1);

StylusPointCollection point = new StylusPointCollection(ptlist);

Stroke stroke = new Stroke(point)

{

DrawingAttributes = inkCanvasMeasure.DefaultDrawingAttributes.Clone()

};

inkCanvasMeasure.Strokes.Add(stroke);

## 全局异常捕获

protected override void OnStartup(StartupEventArgs e)

｛

this.DispatcherUnhandledException += App\_DispatcherUnhandledException;

TaskScheduler.UnobservedTaskException += TaskScheduler\_UnobservedTaskException;

AppDomain.CurrentDomain.UnhandledException += CurrentDomain\_UnhandledException;

｝

void App\_DispatcherUnhandledException(object sender, System.Windows.Threading.DispatcherUnhandledExceptionEventArgs e)

{

try

{

e.Handled = true;

MessageBox.Show("捕获未处理异常:" + e.Exception.Message);

}

catch (Exception ex)

{

MessageBox.Show("程序发生致命错误，将终止，请联系运营商！");

}

}

void CurrentDomain\_UnhandledException(object sender, UnhandledExceptionEventArgs e)

{

StringBuilder sbEx = new StringBuilder();

if (e.IsTerminating)

{

sbEx.Append("程序发生致命错误\n");

}

sbEx.Append("捕获未处理异常：");

if (e.ExceptionObject is Exception)

{

sbEx.Append(((Exception)e.ExceptionObject).Message);

}

else

{

sbEx.Append(e.ExceptionObject);

}

MessageBox.Show(sbEx.ToString());

}

void TaskScheduler\_UnobservedTaskException(object sender, UnobservedTaskExceptionEventArgs e)

{

MessageBox.Show("捕获线程内未处理异常：" + e.Exception.Message);

e.SetObserved();

}

## COM组件使用

右键解决方案增加 用户控件（非wpf那个）UserControl1

工具栏选择项增加WMP控件，将控件拖入UserControl1的窗口

UserControl1的cs文件中加入接口函数

public void play(String mediaPathName)

{

this.axWindowsMediaPlayer1.URL = mediaPathName;

}

调用：

xaml中拖入 WindowsFormsHost控件

标签加入Name属性为host1

UserControl1 f = new UserControl1();

host1.Child = f;

[f.play("D:\\xxx.wmv");](mailto:f.play(@%22D:/xxx.wmv%22);)

## 定时器

using System.Windows.Threading;

DispatcherTimer m\_timer = new DispatcherTimer();

m\_timer.Interval = TimeSpan.FromMilliseconds(1000);

m\_timer.Tick += new EventHandler(m\_timer\_Tick);

m\_timer.Start();

## 消息处理

this.KeyUp += MainWindow\_KeyUp;

键盘消息

if (e.Key == Key.Up)

Key.OemPlus【+键】

Key.OemMinus【-键】

Key.Return【回车】

## 接收/截获消息

using System.Windows.Interop;

///添加函数

IntPtr hwnd = new WindowInteropHelper(this).Handle;

HwndSource.FromHwnd(hwnd).AddHook(new HwndSourceHook(fun));

private IntPtr fun(IntPtr hwnd, int msg, IntPtr wParam, IntPtr lParam, ref bool handled)

{// uint l = (uint)lParam;可以将lParam转成数字

if (msg == 0x0400+1){ }

return IntPtr.Zero;

}

## 命令行参数

App.xaml中加入属性Startup="Application\_Startup"

在App.xaml.cs生成的函数中

private void Application\_Startup(object sender, StartupEventArgs e)

{

if (e.Args.Length == 0)

return;

foreach (string arg in e.Args){}

}

## 动画

using System.Windows.Media.Animation;

/////////////////设置控件在1s内从不透明变成半透明

Storyboard story = new Storyboard();

DoubleAnimation animation = new DoubleAnimation();

//form和to是指代属性变化前后的值

animation.From = 1;

animation.To = 0.5;

animation.Duration = TimeSpan.FromMilliseconds(1000);

Storyboard.SetTarget(animation, this.button1);

//单一属性名可以通过控件的属性列表查找

Storyboard.SetTargetProperty(animation, new PropertyPath("Opacity"));

story.Children.Add(animation);

story.Completed += Story\_Completed; //动画完成完成时的触发事件，在开始动画前设置

story.Begin();

## 右键菜单

控件带context属性

<ListBox.ContextMenu>

<ContextMenu Name="cm" StaysOpen="true">

<MenuItem Header="选择文件夹"/>

<MenuItem Header="一级菜单">

<MenuItem Header="二级菜单"/>

</MenuItem>

</ContextMenu>

</ListBox.ContextMenu>