# WPF经典编程实例

## 在Visual Studio 2019中安装WPF工作负载(Workload)

1. 课程回顾
   1. 《Microsoft Blend 2019》
2. Visual Studio 下载地址：
   1. <http://www.microsoft.com>
3. Visual Studio 安装和配置
4. 课程源代码
   1. <https://github.com/zmrbak/WpfClassicalExamples>

## 创建一个新的WPF应用程序

1. Visual Studio 2019 启动页面
   1. VS中“打开最近使用的内容”
   2. Windows图标中“最近”
2. “克隆或迁出代码”
   1. 克隆<https://github.com/zmrbak/WpfClassicalExamples>
   2. 如果登录，则只能克隆自己的项目
   3. 克隆一个项目
3. “打开项目或解决方案”
4. “打开本地文件夹”
5. “创建新项目”
6. “继续但无需代码”
   1. 在纯粹地启动VS时显示
7. 创建新项目
8. 项目中文件介绍
   1. App.config
   2. 遇到不熟悉的地方，按F1
   3. MainWindow
   4. 代码隐藏类

修改界面开始路径/App.xaml属性



窗口图标/Window属性

Window属性

**单个界面icon**



**所有界面统一icon**

图标必须是.ico，单个window的icon 优先级大于 解决方案中WpfApp1的属性的图标优先级





icon：设置窗口图标

ResizeMode：

NoResize：无最大最小化按钮

CanMinimize：有最小化按钮

CanResize：有最大最小化按钮

CanResizeWithGrip：有最大最小化按钮

但是右下角有网格图案

ShowInTaskbar：

勾选-->打开窗口时电脑显示任务栏图标

不勾选-->不显示

SizeToContent：窗口是否根据宽高自动调整大小

Title：窗口标题

Topmost：打勾 代表该窗体始终处于其它页面的顶端

WindowStartupLocation：界面刚打开时位于电脑屏幕的位置

WindowState：界面打开时是正常、最大化还是最小化状态

Cursor：鼠标的显示状态

## 如何从窗口(Window)中创建新窗口

1. 创建新项目
2. 窗口介绍
3. 创建第二个窗口
4. 使用Button调用第二个窗口
5. 模态和非模态 -- **.show( ) 和 .showDialog( )**
6. 继承于Window的窗口的属性
7. 继承于Window的窗口的方法

## 如何创建页面(Page)和在页面之间导航

1. 基于浏览器风格的导航机制
2. 可用在独立的应用程序中
3. 可用在浏览器应用程序中(几乎没用)
4. 被NavigationWindow和Frame所驻留
5. 创建一个WPF项目(Frame/NavigationWindow)，添加Page
6. HyperLink

## 如何创建一个对话框（Dialog）

1. 自定义窗口(Window)作为对话框
2. 自定义窗口对话框的常用设置
3. 获取自定义对话框的返回值
4. 打开文件对话框
5. 保存文件对话框
6. 打印文件对话框
7. 字体选择对话框
8. 颜色选择对话框

## 如何在窗口(Window)之间建立所属关系

1. 使用VS演示所属关系
2. 创建两个窗口，演示
3. 建立两个成窗口之间的所属关系
4. 修改App.xaml和App.xaml.cs –设置Owner

## 如何创建一个单实例的WPF应用程序

1. 为何要限制启动多个应用程序
2. 如果不想让用户创建多个实例，应该安静地退出，并把已经启动的程序放到前台。
3. 使用Mutex完成互斥
4. 使用Win32 API把已经启动的程序放到前台。
5. FindWindowW、SetForegroundWindow

//\*\*\*目的🡪在bin/Debug中点击应用程序，多次点击也只显示一个，并且在其他程序覆盖时点击会显示出来

## 如何向WPF应用程序传递启动参数

## 如何捕捉WPF应用程序中未处理的异常

// 处理程序中所有的异常

//不知道发生什么异常的时候，如果不解决异常可能会导致程序崩溃，

//可以在App.xaml.cs中解决

//记录日志，记录发生什么异常

//如果程序关闭或者是内部逻辑错误可以在此处重启程序

## 如何使用TextBlock控件向UI中添加纯文本

////-多行，加粗，斜体，截断，超链接等等

## 如何通过Label向其它控件添加文本内容

**<!--Label里可以加其他控件-->**

**<!--Label与TextBlock最大区别是:Label可以涉及热点、快捷键，下例 -->**

<!--让Label和TextBox产生关系-->

//选择Label》》属性》》杂项》》Target》》创建数据绑定》》绑定类型》》ElementName》》选择要绑定的元素名》》确定

<!--绑定完，最后在Label的Content="UserName"的前面加下划线\_,运行不显示下划线，按Alt首字母有下划线，按Alt+U，光标移动到指定位置-->

**\*\*\*\* TextBlock是轻量级，系统负载更小，功能小于Label，如果涉及热点、快捷键使用Label，不然一般使用TextBlock**

## 如何为用户提供一个输入文本的可能性（TextBox）

1. 单行
2. 默认内容、输入内容
3. 样式设置 背景色、前景色等
4. 只读 <!--只读 >> 属性》》文本》》勾选IsReadOnly-->
5. 禁用 <!--禁用（置灰色） >> 属性》》公共》》不勾选IsEnabled-->
6. 多行（类似于文本编辑器）<!--多行 >> 属性》》文本》》勾选AcceptsReturn 代表允许回车-->
7. 滚动条 <!--滚动条 >> 属性》》布局》》

VerticalScrollBarVisibility垂直滚动条(auto文本超过高度才出现滚动条)

HorizontalScrollBarVisibility水平滚动条-->

1. 对于Windows剪贴板的支持

## 如何为WPF应用程序添加图片（Image）

1. Image控件，四种模式
2. 旋转
3. 在代码中为Iamge的Source其赋值
4. BitmapImage

几种添加图片的方法：

Xaml添加图片

方法一：

<!—选择Image》》属性》》公共》》Source-->

四种模式：<!--依次更改》》属性》》公共》》Stretch以及StretchDirection-->

方法二：优点：可以设置旋转

<!--通过Xaml代码设置图片-->

<Image>

<Image.Source>

<!--属性》》杂项》》UriSource 设置图片 ，Rotation 设置 旋转角度 -->

<BitmapImage UriSource="/picture1.jpg" Rotation="Rotate180"/>

</Image.Source>

</Image>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

手动添加图片

方法一：缺点：需更改图片属性，并且和xaml的相同图片冲突，需要复制2份图片

imageA.Source = new ImageSourceConverter().ConvertFromString("picture2.jpg") as ImageSource;

//\*\*右键图片文件》》属性》》高级 -- 复制到输出目录：不复制 改为 始终复制 且 生成操作：Resource 改为 无

方法二：**优点：不需要设置图片属性！！！**

imageA.Source = new BitmapImage(new Uri("picture1.jpg", UriKind.RelativeOrAbsolute));

## 如何在WPF中直接使用2D图形

1. 矩形 Rectangle
2. 椭圆 Ellipse
3. 直线 Line
4. 折线 Polyline
5. 多边形 Polygon
6. 路径 Path
7. 标记语言—M10,20 C20,10之类的

<https://docs.microsoft.com/en-us/dotnet/framework/wpf/graphics-multimedia/path-markup-syntax>

1. 使用XAML代码画一个复杂的图形

## 如何使用ToolTip让控件显示附加的信息

1. 参考VS进行效果演示
2. 创建按钮，为其设置ToolTip
3. 禁用某按钮，观察TooTip
4. 关于ToolTip显示的时间长短
5. 非Button控件同样适用（TextBlock）
6. 偏移
7. 一个复杂的TooTip

**方法一：<!--属性》》公共》》ToolTip-->**

<!--在IsEnabled="False"禁用按钮的时候也显示ToolTip ToolTipService.ShowOnDisabled="True" -->

<!--自定义ToolTip显示的时间 ToolTipService.ShowDuration="2000" 🡪 2000毫秒=2秒 ,

水平偏移 ToolTipService.HorizontalOffset ="100" -->

**方法二：<!--自定义ToolTip-->**



<Button Content="保存" Width="100" Height="30" Margin="20" >

<Button.ToolTip>

<StackPanel>

<TextBlock Text="保存文件" FontWeight="Bold"/>

<TextBlock Text="点击这个按钮，保存这个文件" />

<!--画一条线用Border，BorderThickness：左，上，右，下-->

<Border BorderBrush="Silver" BorderThickness="0,1,0,0" Margin="0,4"/>

<TextBlock Text="按F1获取帮助信息（F1）" FontStyle="Italic"/>

</StackPanel>

</Button.ToolTip>

</Button>

## 如何为WPF应用程序添加Windows标准菜单（Menu）

1. Windows标准菜单
2. 添加菜单 <Menu/> 属性》》公共》》Items，Windows标准是添加MenuItem
3. 添加菜单项 <MenuItem/>
4. 关于分割线 <Separator/>
5. 菜单项中的图标
   1. <https://www.iconfont.cn/>

<MenuItem.Icon>

<!--属性》》公共》》Source，图标和打勾互斥-->

<Image Source="/咖啡.png"/>

</MenuItem.Icon>

1. Unicode字符图标
   1. <http://www.htmleaf.com/ziliaoku/qianduanjiaocheng/20141225979.html>
   2. Unicode字符颜色 -- <MenuItem Header="关闭(\_C)" Icon="&#x2764;" Click="MenuItem\_Click"/>
2. 菜单项中的选择标记 – 是否✔ --IsCheckable="True"
3. 菜单项的快捷键 字符前加下划线 \_
4. 菜单项的键盘访问热键
5. 菜单项点击事件处理

//退出程序

Environment.**Exit**(0);

## 如何为WPF应用程序添加上下文菜单(右键得到的上下文菜单)

1. 上下文菜单ContextMenu
2. 参考上一课所讲的Windows标准菜单Menu

<!--属性》》杂项》》ContextMenu》》新建--> 后续和16课一致

<TextBlock>

## 如何在WPF中为用户提供单选和多选的功能

1. RadioButton
2. CheckBox
3. Group

<!--总结：如果有组名GroupName，组名相同的互斥，剩余的相互互斥。

如果没有组名，以父控件分组，组内互斥-->

-属性》》公共》》勾选IsThreeState 实现RadioButton的三态

//三选 //状态逻辑 true false null

isChecked – 是否被选中

## 如何使用进度条（ProgressBar）控件

1. 进度条值的改变 Value
2. 进度条值不确定 IsIndeterminate="True"
3. 最大值，最小值，值Value。进度条增加、减少。开始、关闭进度条

## 如何使用滑块来选择一个数字（Slider）

1. Windows音量
2. 网易云音乐

## 如何在WPF应用程序中使用日历控件（Calendar）

1. Windows日历

Wpf设置SelectionMode -- 单选 多选 间隔多选显示时间 sender和e的取用

## 如何在WPF应用程序中使用列表框控件（ListBox）

1. Word中的字体列表框
2. Word中的编号列表框

单选-多选-拓展多选-加图标-添加删除

## 如何在WPF应用程序中使用下拉选项框（ComboBox）

1. Word中的字体列表框

## 如何在WPF应用程序使用状态栏（StatusBar）

手写模板 – 状态-分隔-进度条--

## 如何在WPF应用程序中使用工具栏面板（ToolBarTray）

一般在菜单栏下面并且在窗口顶部 – 放Word 字体加粗，斜体，下划线，字号大小

## 如何使用Gird来构建UI布局

1. 平均分配
2. 固定尺寸
3. \*
4. Auto
5. 可手动调整大小 <GridSplitter/>
6. 行扩张 Grid.RowSpan
7. 列扩展 Grid.ColumnSpan

<Separator Height="5" Background="Red"/> 分隔控件

## 如何使用UniformGrid来摆放子控件

-----使行列等分

1. 自动排列子控件
2. 添加子控件
3. 手工设置行数Columns="4"
4. 手动设置列数Rows="2"
5. 手动设置起始列数 FirstColumn="2"
6. 反方向排列 FlowDirection="RightToLeft"

<UniformGrid Columns="4" FirstColumn="2" FlowDirection="RightToLeft">

<Label Content="单元格1" Background="Red"/>

<Label Content="单元格2" Background="Green"/>

<Label Content="单元格3" Background="Blue"/>

<Label Content="单元格4" Background="Yellow"/>

<Label Content="单元格5" Background="Black"/>

<Label Content="单元格6" Background="Beige"/>

<Label Content="单元格7" Background="DarkBlue"/>

</UniformGrid>

## 如何使用WrapPanel自动摆放子控件

1. 大小不等的子控件
2. 子元素控件默认大小

类似于BootStrap

Orientation="Vertical" – 从下往上拉自动排列

Orientation="Horizontal" – 从右往左拉自动排列

ItemHeight="80" – 设置所有控件统一高度-

## 如何在StackPanel中摆放子控件（常用！！！）

1. Horizontal
2. Vertical
3. HorizontalAligment
4. VerticalAligment

StackPanel的 Orientation="Horizontal" – 控件的 VerticalAligment有效

Vertical HorizontalAligment

默认值为拉伸

WrapPanel换行，StackPanel不换行，

程序的性能高于Grid，尽可能用StackPanel替代Grid

## 如何在画布(Canvas)中摆放子控件

1. 使用精确的坐标 Canvas.Left ↑ ↓ ← →
2. 子控件的两个属性 VerticalAlignment和 HorizontalAlignment 不生效
3. Z-Order Panel.ZIndex="1"
4. 通常情况下不使用，画图下会使用

## 如何使用Border来包装UI元素

常用：

BorderThickness 边框粗细 – 所有、左右 上下 、 左 上 右 下

BorderBrush 边框颜色

CornerRadius 弧度

Background 背景色

属性》》画笔/外观 等

## 如何在WPF应用程序中使用滚动面板（ScrollViewer）

<!--属性》》布局》》-->

1. 水平滚动条 HorizontalScrollBarVisibility
2. 垂直滚动条 VerticalScrollBarVisibility

## 如何使用WPF中最常用的控件DockPanel

停靠面板

控件设置 DockPanel.Dock

先设置的先显示全面

<DockPanel>--属性 – 布局 – LastChildFill 最后一个控件是否填充，一般为填充状态

## 如何使用ViewBox来缩放UI元素

自动适应屏幕分辨率 基于窗口大小缩放内容大小

<!--属性》》外观-->

Stretch ， StretchDirection

一个ViewBox只能放一个控件





## 如何使用TabControl控件创建选项卡布局

<!--属性》》公共》》IsSelected-->

<!--手动美化Header-->

## 如何在面板中动态添加/删除子元素

本例：在画布上 左键添加矩形，右键删除矩形。

## 如何在WPF应用程序中提供拖放功能

本例：左边的元素拖动到右边

WrapPanel <!--属性》》公共》》允许拖动AllowDrop-->

## 如何使用CLR属性以及UI通知（重点）

InotifyPropertyChanged

Command绑定

## 如何使用WPF中的依赖项属性（重点！！！）

WPF中大量使用依赖项属性，耗内存少，访问速度快！

//propdp

Command绑定

## 如何使用WPF中的附加属性

## 如何在WPF中把数据绑定到对象

## 如何在WPF中把数据绑定到集合

## 如何使用元素与元素之间的数据绑定

## 如何在DataGrid中对数据进行排序

## 如何在DataGrid中对数据进行分组

## 如何在DataGrid中对数据进行筛选

## 如何在WPF中使用静态绑定

## 如何在WPF中使用值转换器

## 如何在WPF中使用多值转换器

## 如何在WPF中创建和使用自定义控件

## 如何在WPF中定制自定义控件

## 如何在自定义控件中公开属性

## 如何在自定义控件中公开事件

## 如何使用行为来扩展控件的功能

## 如何在WPF中创建和使用用户控件

## 如何在用户控件中公开事件

## 如何为控件创建和使用样式

## 如何基于样式创建新的样式

BaseOn

## 如何为控件自动应用样式

## 如何编辑现有控件的模板

## 如何为控件创建属性触发器

## 如何为控件创建多属性触发器

## 如何创建底层数据的触发器

## 如何创建底层多数据的触发器

## 如何创建和使用事件触发器

## 如何在WPF应用程序中使用二进制资源

## 如何在WPF中使用其它程序集中的二进制资源

## 如何在WPF的隐藏代码中访问二进制资源

## 如何在WPF中使用静态逻辑资源

## 如何在WPF中使用动态逻辑资源

1. StaticResource 一次性绑定
2. DynamicResource 在需要的时候绑定

## 如何有效管理逻辑资源

## 如何使用用户为系统选择的颜色和字体

## 如何使用MVVM模式来构建WPF应用程序

## 如何在WPF应用程序中使用路由命令

## 如何在渲染时缩放元素（ScaleTransform）

1. ScaleTransform
2. ScaleX
3. ScaleY
4. CenterX、CenterY

## 如何在渲染的时候旋转元素（RotateTransform）

1. RotateTransform
2. Angle
3. CenterX、CenterY

## 如何在渲染的时候倾斜元素（SkewTransform）

1. SkewTransform
2. AngleX、AngleY
3. CenterX、CenterY

## 如何在渲染的时候移动元素（TranslateTransform）

1. TranslateTransform
2. X
3. Y

## 如何对多个转换进行分组（TransformGroup）

1. RenderTransformOrigin
2. TransformGroup

## 如何创建基于属性的动画

1. System.Windows.Media.Animation
2. System.Windows.Media.Animation.Animatable.Timeline

## 如何创建基于路径的动画

1. PathGeometry

## 如何创建基于关键帧的动画

## 如何为动画添加缓动效果（Easing Effects）

1. Beizer

## 如何创建一个WCF服务

## 如何创建一个自托管的WCF服务

1. System.ServiceModel.ServiceHost

## 如何在IIS中托管WCF服务

1. Windows Server
2. Windows 10
   1. Win10安装IIS并配置ASP.NET 4.0
   2. 添加MIME类型：扩展名“.svc”，MIME类型 “application/octet-stream”
   3. 在“Handler Mappings(处理程序映射)”中添加Managed Handler(添加托管处理程序)

Request path: \*.svc

Type: System.ServiceModel.Activation.HttpHandler

Name: svc-Integrated

* 1. 添加system.serviceModel
  2. 删除 serviceHostingEnvironment

1. inetmgr

## 如何在WPF应用程序中集成WCF服务

## 如何为XAML 启用UI调试工具

## 如何使用实时可视化树在XAML元素中导航

## 如何使用实时属性资源管理器检查XAML属性

## 如何从非UI线程中更新UI元素

## 如何对长时间执行的线程提供一个取消的功能

## 如何使用Backgroundworker组件

1. System.ComponentModel.BackgroundWorker

## 如何使用定时器来周期性更新UI界面

1. DispatcherTimer

## 如何在WPF应用程序中驻留WinForm控件

1. System.Windows.Forms.Integration.WindowsFormsHost
2. 在WPF中驻留一个或者多个WinForm控件
3. 在WPF元素中驻留一个或者多个复合控件
4. 在WPF中驻留一个或者多个ActiveX控件
5. 一个使用WPF、另一个使用WinForm相互交互的两个控件
6. 不支持多级混合控件

## 如何在WinForm应用程序中驻留WPF控件

1. PresentationCore
2. PresentationFramework
3. System.Xaml
4. UIAutomationProvider
5. WindowsBase
6. WindowsFormsIntegration
7. WpfControlLibrary1

## 如何在WPF应用程序中调用WIN32API

1. <http://www.pinvoke.net>
2. SetForegroundWindow

## 如何在WPF应用程序中嵌入ActiveX控件

1. 命令：aximp
2. Developer Command Prompt for VS 2019 Preview

## 如何从A窗口中更新B窗口中的内容

## WPF如何将DLL嵌入到exe文件中

1. 使用第三方库的WPF应用程序
2. 修改项目文件

|  |
| --- |
| <Target Name="AfterResolveReferences">  <ItemGroup>  <EmbeddedResource Include="@(ReferenceCopyLocalPaths)" Condition="'%(ReferenceCopyLocalPaths.Extension)' == '.dll'">  <LogicalName>%(ReferenceCopyLocalPaths.DestinationSubDirectory)%(ReferenceCopyLocalPaths.Filename)%(ReferenceCopyLocalPaths.Extension)</LogicalName>  </EmbeddedResource>  </ItemGroup>  </Target> |

1. 修改App.Xaml.cs

|  |
| --- |
| private static Assembly OnResolveAssembly(object sender, ResolveEventArgs args)  {  Assembly executingAssembly = Assembly.GetExecutingAssembly();  var executingAssemblyName = executingAssembly.GetName();  var resName = executingAssemblyName.Name + ".resources";  AssemblyName assemblyName = new AssemblyName(args.Name); string path = "";  if (resName == assemblyName.Name)  {  path = executingAssemblyName.Name + ".g.resources"; ;  }  else  {  path = assemblyName.Name + ".dll";  if (assemblyName.CultureInfo.Equals(CultureInfo.InvariantCulture) == false)  {  path = String.Format(@"{0}\{1}", assemblyName.CultureInfo, path);  }  }  using (Stream stream = executingAssembly.GetManifestResourceStream(path))  {  if (stream == null)  return null;  byte[] assemblyRawBytes = new byte[stream.Length];  stream.Read(assemblyRawBytes, 0, assemblyRawBytes.Length);  return Assembly.Load(assemblyRawBytes);  }  }  protected override void OnStartup(StartupEventArgs e)  {  base.OnStartup(e);  AppDomain.CurrentDomain.AssemblyResolve += OnResolveAssembly;  } |