XAML文档定义了在WPF应用程序中组成窗口的面板、按钮以及各种控件的布局。

XAML是WPF技术中专门用于设计UI的语言。

程序设计要“高内聚-低耦合”。

XAML是一种由XML派生而来的语言。

XAML中使用标签声明一个元素，每个元素对应内存中的一个对象

非空标签：<Tag Attribute1 = Value1 Attribute2 = Value2>Content</Tag>

空标签：<Tag Attribute1 = Value1 Attribute2 = Value2 />

xmlns：名称空间

xmlns[:可选的映射前缀]=”名称空间”

如果没有前缀，就是默认名称空间，默认名称空间只能有一个。

<Window x:Class="xmal\_test1.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>

</Grid>

</Window>

上例中的XAML，<Window>和<Grid>都来自第二行的默认名称空间，而<Window>中的Class特征则来自第三行中的前缀为x的名称空间。

XAML文档的树形结构：

XAML中为对象属性赋值：

XAML中的一个标签的Attribute（特征）里有一部分与对象的Property互相对应。

例：程序xaml\_test2

<Window x:Class="xaml\_test2.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">

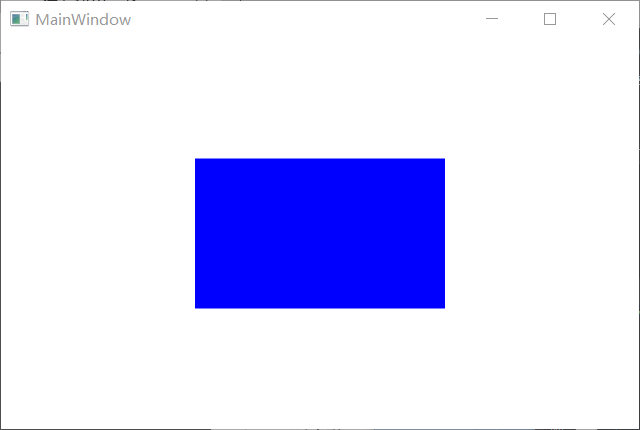
<Grid VerticalAlignment="Center" HorizontalAlignment="Center">

<Rectangle x:Name="rectangle" Width="200" Height="120" Fill="Blue"/>

</Grid>

</Window>

其中，<Rectangle>的标签的Fill特征初始值赋为Blue。效果如下：



通过Attribute=Value赋值时，Value只能是字符串值。这样会带来很大的不便。

使用属性元素为对象赋值

例：程序xaml\_test3

<Window x:Class="xaml\_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 VerticalAlignment="Center" HorizontalAlignment="Center">

<Rectangle x:Name="rectangle" Width="200" Height="120">

<Rectangle.Fill>

<LinearGradientBrush>

<LinearGradientBrush.GradientStops>

<GradientStopCollection>

<GradientStop Offset="0.2" Color="LightBlue" />

<GradientStop Offset="0.7" Color="Blue" />

<GradientStop Offset="1.0" Color="DarkBlue" />

</GradientStopCollection>

</LinearGradientBrush.GradientStops>

</LinearGradientBrush>

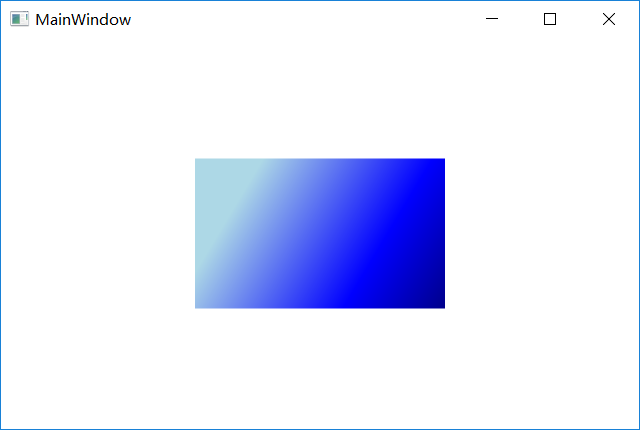
</Rectangle.Fill>

</Rectangle>

</Grid>

</Window>

效果为：



标记扩展：

例：程序xaml\_test4

<Window x:Class="xaml\_test4.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>

<StackPanel Background="LightSlateGray">

<TextBox Text="{Binding ElementName=slider, Path=Value, Mode=OneWay}" Margin="5">

</TextBox>

<Slider x:Name="slider" Margin="5" />

</StackPanel>

</Grid>

</Window>

其中，Text="{Binding ElementName=slider, Path=Value, Mode=OneWay}"

就是标记扩展。有点类似C#中的对象初始化器。

只有MarkupExtension类的派生类才可以使用标记扩展。

当一个XAML标签对应着一个对象时，标签的一部分Attribute对应一个对象的Property，还有一部分Attribute对应着对象的事件。例如<Button>标签有一个名为Click的Attribute，它对应Button类的点击事件。

在.NET中，可以为对象的某个事件指定一个事件处理器（Event Handle），当事件发生时，.NET会调用这个事件处理器。

例：程序xaml\_test5

<Window x:Class="xaml\_test5.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 VerticalAlignment="Center">

<Button Name="ok" Click="ButtonClick" Width="50" Height="50">

</Button>

</Grid>

</Window>

其中<Button>标签的事件处理器在MainWindow.xaml.cs中实现。

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace xaml\_test5

{

/// <summary>

/// MainWindow.xaml 的交互逻辑

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

public void ButtonClick(object sender, RoutedEventArgs e)

{

MessageBox.Show("Ok");

}

}

}

WPF编程中，XAML表示UI，C#的代码用于处理程序的逻辑，将逻辑代码与UI代码分离，隐藏在UI代码之后的形式叫代码后置。

大多数情况下，一个完整的程序会被分成若干个模块，每个模块都能独立编译、升级等。.NET的模块叫程序集（Assembly）。常见的程序集以.exe或.dll为扩展名。大多数情况下，引用程序集说的都是动态链接库.dll，所以称为引用类库。

在XAML中，引用类库：