**3.Binding绑定**

Binding绑定是WPF特有的数据自动更新机制。绑定告诉WPF，当源对象的属性值变化时，WPF自动更新目标对象的属性。

# Binding需要哪些配置？

* 目标对象Target
* 源对象Source
* 绑定路径Path
* 绑定模式BindingMode（默认值Defalut，可选）
* 更新源对象属性触发模式UpdateSourceTrigger（默认值Defalut,可选）
* 更新触发延迟Delay（默认值0,可选）

# 目标对象和源对象

在绑定机制中，目标对象和源对象的关系如下：



我们可以看到，目标对象的属性必须是依赖项属性，因为依赖项属性在值发生变化时有更改通知的功能，而Bingding机制依赖于自动更改通知的功能。而且，目标对象必须是继承自依赖对象类型DependencyObject。源对象及其属性可以是任意数据类型。即绑定可以是目标对象的依赖属性绑定到源对象的属性，也可以是目标对象的依赖属性绑定到源对象本身。以下的讲解是基于目标对象的依赖属性绑定到源对象的属性而言。

# 绑定模式

WPF的绑定模式有如下的5种，其中的Default模式取决于目标对象的依赖项属性的元数据FrameworkPropertyMetadata中对BindsTwoWayByDefault属性的设置。

## TwoWay

当源属性（或者目标属性）变化时，目标属性（或者源属性）会自动更新。

## OneWay

当源属性变化时，目标属性会自动更新。

## OneTime

根据最初的源属性设置目标属性，即更新只有一次(除非显式调用BindingExpression.UpdateTarget()方法强制更新)。该模式在源属性固定不变的情况下可降低运行开销。

## OneWayToSource

当目标属性变化时，源属性会自动更新。

## Default

当目标对象的依赖项属性的元数据FrameworkPropertyMetadata中对BindsTwoWayByDefault属性设置为True时，绑定模式将是TwoWay，否则，绑定模式是OneWay。

# 更新源对象属性触发模式

当绑定模式为TwoWay或者OneWayToSource时，目标对象的依赖属性发生变化时，源对象的属性将可能立即发生变化。更新源属性的动作时效性取决于更新源触发模式和更新触发延迟设置，以下共4中模式（假设更新触发延迟Delay未设置，默认毫秒数为0）：

## PropertyChanged

当目标属性发生变化时，源属性立即更新。

## LostFocus

当目标属性发生变化且目标对象失去焦点时，源属性才更新。

## Explicit

唯有显式调用绑定表达式方法BindingExpression.UpdateSource()才能更新源属性。

## Default

Default模式对应的实际更新模式取决于目标对象的依赖项属性的元数据FrameworkPropertyMetadata中对DefaultUpdateSourceTrigger属性的设置，其值可以是PropertyChanged、LostFoucs、Explicit。

# 更新触发延迟

Bingding对象的Delay属性单位为毫秒，默认值为0。我们可以设置Delay值来延迟触发更新源的动作。在绑定输入框的Text属性时，该延迟十分有意义，只有在用户停止输入（触发更新源模式为PropertyChanged）后延迟规定毫秒数才触发更新，从而避免因不停输入键值而频繁更新。

# 如何创建绑定？

绑定可以直接在XAML文件的元素标记中创建，也可以使用代码创建。下面介绍这两种方法的最简单的绑定用法：

## 使用XMAL创建绑定

<TextBlock x:name="tblock" Margin="3" Text="文字大小被Slider控件绑定控制" FontSize="{Binding ElementName=slider, Path=Value, Mode=TwoWay,

UpdateSourceTrigger=PropertyChanged, Delay=1000}"></TextBlock>

<Slider x:Name="slider" Margin="3" TickFrequency="1"

IsSnapToTickEnabled="True" Minimum="1" ></Slider>

上面的XAML标记以TextBlock为目标对象，属性Text为目标属性，而绑定源对象为Slider，源属性为Slider的Value属性。而可选项Mode、UpdateSourceTrigger和Delay也相应设置了值。

## 使用代码创建绑定

上面XAML标记绑定等效于下面的代码：

private void Bind()

{

Binding binding = new Binding();

binding.Source = slider;

binding.Path = new PropertyPath(Slider.ValueProperty);

binding.Mode = BindingMode.TwoWay;

binding.UpdateSourceTrigger = UpdateSourceTrigger.PropertyChanged;

binding.Delay = 1000;

BindingOperations.SetBinding(tb, TextBox.FontSizeProperty, binding);

//tb.SetBinding(TextBox.FontSizeProperty, binding);

}

实际上，tb.SetBinding()方法是对BindingOperations.SetBinding()方法的封装：

public void SetBinding(DependencyProperty dp, BindingBase bind)

{

BindingOperations.SetBinding(this, dp, bind);

}

# 如何取消绑定？

## 移除目标对象的某个依赖项属性的绑定表达式

BindingOperations.ClearBinding(tb, TextBox.FontSizeProperty);

## 移除目标对象的所有依赖项属性绑定表达式

BindingOperations.ClearAllBindings(tb);

# 绑定非元素对象