

AicStudio 用户手册

时间: 2014.12.23 版本: 1.0 编辑: 董松涛,何沛华 邮箱: dongsongtao@aic-tech.com

目录

1	AicStudio 简介	1
	1.1AicStudio 功能介绍	1
	1.2AicStudio 平台特性	1
2	菜单与窗口	5
	2.1 AicStudio 的菜单	5
	2.1.1 文件菜单	5
	2.1.2 主页菜单(Home)	5
	2.1.3 工程菜单(Project)	6
	2.1.4 视图菜单(View)	6
	2.2 AicStudio 的窗口	7
3	工程管理	.11
	3.1 工程创建	.11
	3.2 文件添加	.12
	3.3 程序集引用	.13
4	界面编辑	.14
	4.1 控件 Controls	.14
	4.2 数据绑定 DataBinding	.16
	4.2.1DataContext 绑定	.16
	4.2.2 StaticResource 绑定	.18
	4.2.3 Element Name	.20
	4.2.4 OPC UA Data Node	.23
	4.2.5 OPC UA Relative Node	.25
	4.2.6 相对源绑定 RelativeSource	.31
	4.3 动画 Animation(行为)	.34
	4.3.1 旋转(RotateAction)	.34
	4.3.2 缩放(ScaleAction)	.39
	4.3.3 尺寸变化(SizeAction)	.40
	4.3.4 倾斜扭曲(SkewAction)	.41
	4.3.5 平移(TranslateAction)	.42
	4.4 特效 Effects(效果)	.42
5	脚本	.55
	5.1 代码编辑器	.55
	5.2 脚本向导	.55
	5.2.1 设置相对绑定对象(Set Relative Binding Context)	.56
	5.2.2 修改变量当前值(Write variable's current value)	.56
	5.2.3 读取变量实时值(Read variable's realtime data)	.61
	5.2.4 订阅变量实时值(Subscribe variable's realtime data)	.63
	5.2.5 读取变量历史数据(Read variable's history data)	.66
	5.2.6 解析节点路径为 ID(Translate path)	.70
	5.2.7 调用方法(Call method)	.73
6	编译与调试	.78
7	快速入门示例	.78

7.1 使用 DataHub 建立对象模型和对象实例	78
7.2 使用 IOServer 为 DataHub 提供数据	82
7.3 使用 AicStudio 构建监控画面	86

1 AicStudio 简介

1.1AicStudio 功能介绍

AicStudio 是艾克信控科技有限公司开发的 SCADA 产品套件 AicVision 的客户端监控画面 组态及编程开发工具,基于微软最新的.NET Framework 4.5 框架和 OPC UA 国际标准(IEC 62541)开发,编程语言为 C#,界面描述语言为 XAML,目前的最新版本为 2.0。

1.2AicStudio 平台特性

AicStudio 具有如下特点:

 AicStudio 提供 WYSIWYG (所见即所得)的图形画面组态编辑功能:从"工具箱" 中拖拽控件到设计面板,然后在"属性"面板中设置控件属性或关联实时变量,即 可完成画面的组态。此外,主编辑区域可以在设计面板和标记文本(XAML)之间进行 切换,对于较资深的程序员用户可以通过直接修改 XAML 标记文本来修改界面。



	luon	zxaml × MainWindow.xaml.cs × App.xaml × 升路贝 × MainWindow.xaml* × MainWindow.xaml × MainWindow.xaml ×						
🕸 Mair	۱Win	dow 🔹						
\$ 1	Ę.	<window< th=""></window<>						
2		xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"						
3		xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"						
4		xmlns:aic="http://www.aic-tech.com/2012/xaml"						
5		xmlns:ee="http://schemas.microsoft.com/expression/2010/effects"						
6		x:Class="WpfApplication2.MainWindow"						
7		Title="MainWindow"						
8		Height="350"						
9		Width="525"						
10		DataContext="100">						
11	皁	<window.resources></window.resources>						
12	曱	<style <="" th="" x:key="stl"></tr><tr><th>13</th><th></th><th>TargetType Resources</th></tr><tr><th>14</th><td></td><td><Setter Prop</td></tr><tr><th>15</th><th></th><th>Value="ddddssst" /></th></tr><tr><th>16</th><th>上</th><th></style>						
17	닏	<style <="" th="" x:key="sstl"></style>						

- 2) AicStudio 同时支持基于 WPF 4.0/4.5 的 C/S 架构工程以及基于 Silverlight 5.0 的跨浏 览器、跨平台的 B/S 架构工程。
- 3) AicStudio 提供常用的图素和控件库,包括仪表盘、开关、趋势曲线、实时报警控件、历史回放控件以及各种常用符号等。此外,AicStudio 还支持第三方提供的WPF/Silverlight标准控件,例如 Telerik、Syncfusion等控件开发商提供的WPF和Silverlight控件。对于有控件开发能力的用户,还可以自行开发符合WPF/Silverlight标准的自定义控件。
- 4) AicStudio 基于微软最新的 WPF/Silverlight 技术的图形系统,所有的画面都是矢量图,可以实现无级的缩放而不会影响显示的效果。并且支持控件的倾斜、旋转、平移、缩放、翻转等多种变换;支持控件的组合和分解操作;支持不受限制的图层。

EFTORE LIVE TV - E EFTERE	上边缘 小平居中 石边缘 上边缘 垂直居中 下边缘	●●● ●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●	◆◆ ◆◆ ◆◆ ◆ ◆ ◆ ◆ ◆ ◆ ◆ ◆ ◆ ◆ ◆ ◆ ◆ ◆ ◆
顺序	对齐	尺寸	翻转

AicStudio 可以很方便的添加多种动画与特效,让你的界面炫酷 起来。

叟索控件		م _
项目	INI ActivateStateAction	^
⊿ 控件	INI BlinkBooleanPropertyAction	
▶ 艾克信控	INI BlinkBrushPropertyAction	
通用	INI BlinkColorPropertyAction	B
面板	III BlinkDoublePropertyAction	(A
样式	INI BlinkInt32PropertyAction	A
行为	INI BlinkPointPropertyAction	14
形状	INI BlinkStringPropertyAction	
が里	INI BlinkVisibilityPropertyAction	
~~来	INI CallMethodAction	
	INI ChangePropertyAction	
▶ 1⊻1直	INI ControlStoryboardAction	
	DataStateBehavior	
	INI DiscreteBooleanPropertyAction	
	INI DiscreteBrushPropertyAction	
	III DiscreteColorPropertyAction	
	INI DiscreteDoublePropertyAction	
	IN DiscreteInt32PropertyAction	*

5) AicStudio 提供基于 OPC UA 国际标准的开放式数据连接,支持任何标准 OPC UA 服务器(例如 AicVision 套件中的实时数据库产品 AicDataHub 服务器)变量的数据绑定:变量可通过组态直接绑定到控件或图素的属性上,运行时自动向实时数据服务器进行数据订阅,无需用户进行额外的编程。在 AicStudio 提供的 OPC UA 服务器窗口中可以连接 OPC UA 服务器,查看 OPC UA 服务器地址空间,选择关联变量等。



6) AicStudio 支持使用 C#进行编程,功能扩展不受限制。AicStudio 拥有友好的 C#代码 编辑环境,智能提示和代码助手功能大大提升了用户的编程体验。



- **7)** AicStudio 的工程是编译执行的,所有的画面和程序在发布之前都会编译成程序集, 运行时无需再进行额外的处理,大大提高了系统运行时的效率。
- 8) AicStudio 支持面向对象/设备的画面组态,这样组态而成的画面是基于对象/设备类型而非对象/设备实例,在运行时可通过脚本实时切换对象/设备实例,画面所显示的数据将会相应动态切换到指定的对象/设备实例。
- 9) AicStudio 采用了与微软 Visual Studio 开发工具完全兼容的工程文件,各自的 WPF 或者 Silverlight 工程无需修改在 2 个平台上运行都没有障碍,在具体项目开发时,可以结合两个平台各自的优点进行快速便捷的开发操作。例如,AicStudio 只提供了 WPF 工程的简单调试跟踪功能,对于 Silverlight 工程则暂未提供调试跟踪功能,因此在需要的时候,用户完全可以利用 Visual Studio 强大的调试功能对 AicStudio 工程(WPF 或 Silverlight)进行深层次的跟踪和调试。

关于相关具体功能的使用,会在后续的章节一一详细进行介绍。

2 菜单与窗口

2.1 AicStudio 的菜单

AicStudio 这个软件有 5 个菜单,分别为文件菜单、主页(Home)、工程(Project)、视 图(View)和工具(Tools),如图 2.1 所示。



图 2.1

2.1.1 文件菜单

文件菜单如图 2.1.1 所示,可用于新建、打开解决方案,添加新工程和已经存在的工程,右 边是最近使用的解决方案(最近的解决方案也可以在窗口中的开始页看到)。

New Solution	Recent Solutions
	1 WpfTest.sln
🚰 Open Solution	2 SilverlightApplication1.sln
Add New Project	
Add Existing Project	
🛃 Save	最近解决方案
🔣 SaveAs	
🚰 Save All	
📄 Close	
Close All Documents	
Close Solution	
Print	
PrintPreview	
冬	2.1.1

2.1.2 主页菜单(Home)

主页菜单如图 2.1.2 所示,包括剪贴板、编辑、查找、书签、编译、语言设置和界面的 主题选择等工具项。



2.1.3 工程菜单(Project)

工程菜单如图 2.1.3 所示,包括整体解决方案编译、当前项目编译,调试运行等工具项。 Home Project View Tools 9 2 D <u>....</u> Build Rebuild Clean Start Start Stop Debugging Without Debugging debugging Build Rebuild Clean Solution Solution Solution Break Continue Step Step Solution Build Project Build Debug/Run

图 2.1.3

2.1.4 视图菜单(View)

视图菜单如图 2.1.4 所示,包括错误列表、输出窗口、属性窗口、解决方案浏览器、搜索结果窗口、工具窗口、大纲窗口、起始页、资源窗口、OPCUA 服务浏览器、本地变量、监视串窗口、调用堆栈窗口、模块窗口、线程窗口、断点窗口。这些开发过程中显示用的窗口的功能和用法都和 Visual Studio 类似。



图 2.1.4

2.2 AicStudio 的窗口

打开 AicStudio,窗口如图 2.2 所示,包括工具窗口、OPCUA 服务浏览器窗口、大纲窗口、 主窗口、信息输出窗口、解决方案浏览器窗口、属性窗口和资源窗口。这些窗口都可以在视 图菜单(View)中打开,如不需要可以关掉。

Tool	s 4 ×	Start Page 🗙			>	Solutio	on Explorer 🕴 🖗
		Name		Modified 🛛 😽	Location	0	0 🖀
	TERT	SilverlightApplication	<u>n1</u>	11/22/2014	C:\Users\near\Documents\AicStudio\Projects\SilverlightApplication1\Sil		
	上共図口和	WpfTest		12/5/2014	$C: \label{eq:series} C: \lab$		初步之安
							脌/犬刀杀
	UFCUA 服务						浏览型
	浏览架容口			主窗[バリソリム合合
	バリル品図ロ	•			• • • • • • • • • • • • • • • • • • •		
		Open Solu	tion	New Solution			
Tools OPC UA Server Explorer							
Outli	Outline 0 ×				Proper	rties 🄱	
l r							
-	大纲窗口	Output			♦ × Error List	1	属性和资
Ľ		Show output from: E	Build 🔻		Project File Line Column Error N		
							酒窗口 しんしょう しんしょう しんしょう しんしょう しんしょう おおし おおし しんしょう おおし おおし しんしょう しんしょ しんしょ
				信息	前出窗口		
					·		
		•					
		Output Locals Thre	eads Breakpoints		Error List Find Results Watch Call stack Modules	Prope	rties Kesources

图 2.2

工具窗口如图 2.3 所示,开发中可以拖拽基本的控件、AicVision 的控件、第三方控件(如 Telerik 控件)、样式,控件动画等,在下文中会具体用到。



图 2.3

OPC UA 服务浏览器窗口如图 2.4 所示,用于添加、删除 OPC UA 服务器,在下文中会用 到。

OPC UA Server Explorer	ů ×
OPC UA Servers	
AicTech DataHub Server	
Tools OPC UA Server Explorer	

图 2.4

大纲窗口如图 2.5 所示,可以看到当前的编辑界面的所有控件和其层级关系,图中的编辑界面可见按钮可以让对应的控件在当前编辑界面中不可见或可见(不会影响控件的实际属性,只是方便用户开发),编辑界面 Lock 按钮用于给对应的控件加锁,加锁后此控件将不可编辑(修改控件的属性)。



图 2.5



图 2.6

信息输出窗口包含以下几个窗口:输出窗口、本地变量窗口、线程窗口、断点窗口、包 括错误列表窗口、搜索结果窗口、监视串窗口、调用堆栈窗口、模块窗口。这些窗口可以用 于调试进行数据的输出。

解决方案浏览器窗口如图 2.7 所示,可看到当前的解决方案下的各个项目以及各个项目 下的各个文件,也可以在此窗口进行项目和文件的添加、编辑、删除等操作。



图 2.7

属性窗口如图 2.8 所示,显示当前选中控件的属性,直观地设置控件的属性、事件、数 据绑定等。

Propertie	es	ů ×
Name	LayoutRoot	
Туре	Grid	\$
E 24		٩
App	earance	-
	Opacity □ (100%	
	Visibility D Visible	~
	Effect 🗆 💌	•
Srus	sh	-
Properti	es Resources	

图 2.8

资源窗口如图 2.9 所示, 层级显示当前选中的控件的 XAML 标记文本中定义过的资源。

Resources	ņ	\times	
 		ф.	
Ф App.xaml			
▲ III MainPage.xaml			
₽ [UserControl]			
R button			
Properties Resources			

图 2.9

3 工程管理

3.1 工程创建

运行 AicStudio 主程序,在主菜单点击 New Solution 新建解决方案。



AicStudio 可以新建的工程类别有三种,WPF、Silverlight 和 Web。可以根据工程需要进行选择,现以 WPF Application 工程为例新建项目。

New Project —				
Categories		Templates		
🔺 🚈 C#		BawPF Application		
— 🧰 Silverlight	t	關WPF Control Library		
🔤 Web				
🔤 WPF				
A project for creatin	ıg rich desktop	applications that run on Windows.		
<u>N</u> ame	WpfApplicati	on2		
Location	C:\Users\dst\	Documents\AicStudio\Projects		
Solution Name WpfApplicat		on2 Create directory for solution		
Framework Version 4.5		•		
		OK Carrol		

新建工程完成后,在屏幕的右上角可以看到解决方案浏览器(Solution Explorer)。我们在这里进行文件的添加、删除,程序集引用的添加和删除等操作。



3.2 文件添加

右键点击工程, 送	选择 Add New Item	选项进行添加新文件。
-----------	-----------------	------------

Solution Explorer		т 🗠
Q 🗗 📴 🚰		
WpfApplication2		
 WpfApplication2 Properties References App.config App.xaml Class1.cs MainWindow: 	Build Rebuild Clean Add Reference Add Service Reference Add Web Reference Add New Item	
■	Add Existing Item Add New Folder Add Existing Folder Set as Startup Project Start Without Debugging Ctrl+F5	

以添加新的类为例,在 General 菜单下选择 Class,输入类名称,点击 OK 进行添加。

New Item	– 🗆 ×
Categories	Templates
⊿ 🦾 C#	Application Configuration File
🔚 General	EApplication Manifest File
- 🔚 Silverlight	Assembly Information File
- WPF	~;)Class
	-••Interface
	🖪 Resource File
	Text File
	💭 Xml File
Create an empty class declaration	l.
Name Class1.cs	
	<u>O</u> K <u>C</u> ancel

3.3 程序集引用

在工程的 References 文件夹下右键点击 Add Reference,添加程序集引用



引用的程序集主要分4类。

1) Framework, 是.NET Framework 框架下的程序集;

2) Liabrary,是扩展或第三方控件库的集合,例如 Telerik, AicVision, ASP.NET, SQLServer 等;

- 3) Browse,通过浏览文件来添加程序集引用;
- 4) COM, 添加 COM 组件。

Edit References	– 🗆 ×
Framework Library Browse COM	Selected References:
Name Version Package	+□ Microsoft.CSharp
EQATEC.Analytics.Monitor 3.2.1.0 RadControls for WPF Telerik.Pivot.Core 2014.1.331.40 RadControls for WPF Telerik.Pivot.DataProviders.Adomd 2014.1.331.40 RadControls for WPF Telerik.Pivot.DataProviders.Queryable 2014.1.331.40 RadControls for WPF Telerik.Pivot.DataProviders.Xmla 2014.1.331.40 RadControls for WPF Telerik.Windows.Controls.Chart 2014.1.331.40 RadControls for WPF Telerik.Windows.Controls.Data 2014.1.331.40 RadControls for WPF Telerik.Windows.Controls.DataServices 2014.1.331.40 RadControls for WPF Telerik.Windows.Controls.DataServices 2014.1.331.40 RadControls for WPF Telerik.Windows.Controls.DataServices 2014.1.331.40 RadControls for WPF Telerik.Windows.Controls.DataVisualization 2014.1.331.40 RadControls for WPF Telerik.Windows.Controls.DataVisualization 20	System Xml System Xml WindowsBase
	<u>O</u> K <u>C</u> ancel

4 界面编辑

界面编辑是软件的 UI 部分,决定软件界面的显示和与用户交互时的行为。在 AicStudio 中,有非常方便的界面编辑工具,下面的小节将从各类控件,数据绑定选择,动画添加等几 个方面进行介绍。

4.1 控件 Controls

亏控件添加之前,	要先打开编辑界面,	即双击 MainWindow.xaml,	显示设计界面
MainWindow.xaml.cs × Main	Window.xaml × Class1.cs ×		×
_			
-	MainWindow		
			-
100% - 🙉 🎟 📾 🔅	₽.		• •

然后可以看到屏幕左上角显示出来的工具箱 Tools,工具箱中包含了可以添加进界面的所有 控件,除了 Common 菜单下的基本控件 Button、Border、Canvas、CheckBox 之外,还有 AicVision 菜单下,艾克信控公司开发的系列控件,在 Locations 菜单下包含所有本地的控件程序集, 常用的如 Telerik 等第三方提供的控件库。

Search Assets	
Project	 AccessText
▲ Controls	🞞 Border
 AicVision 	■ BulletDecorator
Common	🖓 Button
Panels	🛗 Calendar
Styles	🗔 CalendarButton
Behaviors	🗔 CalendarDayButton
Shapes	 CalendarItem
Effects	🔁 Canvas
Symbols	CheckBox
Locations	🔁 ComboBox
Locations	🔁 ComboBoxItem
	ContentControl
	ContentPresenter
	<> Control
	DataGrid
	🔊 DataGridCell
	() DataGridCalleDracantar

Tools X OPC UA Server Explorer X	
Tools	¢ 🗙
Search Assets	٩
Project	🔒 🚺 ArrowButton 🍵
 Controls 	FlatToggleSwitch
 AicVision 	GelButton
Alarm	GelButtonBase
Buttons	GelRepeatButton
Containers	GelToggleButton
Gauges	GenericButton
Indicators	GenericButtonBase
Input	GenericRepeatButton
Labels	GenericToggleButton
LEDDisplays	IndustrialPushButton
Misc	IndustrialRepeatButton
Motors and Fans	IndustrialToggleButton
Pipes	PowerToggleButton
Scrolling	RotarySwitch3State
Sliders	W Switch001
Tanks	witch002
Tools × OPC UA Server Explorer ×	- IBM Suntabring

arc	:h Assets	
	Telerik.Windows.Controls.EntityFramework60.dll	🛅 RadGroupHeader
	Telerik.Windows.Controls.Expressions.dll	RadOrderedWrapPanel
	Telerik.Windows.Controls.FixedDocumentViewer	T RadRibbonBackstage
	Telerik.Windows.Controls.FixedDocumentViewer	RadRibbonBackstageItem
	Telerik.Windows.Controls.GanttView.dll	RadRibbonButton
	Telerik.Windows.Controls.Gauge.dll	RadRibbonComboBox
	Telerik.Windows.Controls.GridView.dll	🖹 RadRibbonComboBoxItem
	Telerik.Windows.Controls.ImageEditor.dll	💾 RadRibbonContextualGroup
	Telerik.Windows.Controls.Input.dll	📼 RadRibbonDropDownButtor
	Telerik.Windows.Controls.Navigation.dll	🛅 RadRibbonGallery
	Telerik.Windows.Controls.Pivot.dll	🛗 RadRibbonGroup
	Telerik.Windows.Controls.PivotFieldList.dll	RadRibbonRadioButton
	Telerik.Windows.Controls.RibbonView.dll	📼 RadRibbonSplitButton
	Telerik.Windows.Controls.RichTextBoxUI.dll	💾 RadRibbonTab
	Telerik.Windows.Controls.ScheduleView.dll	RadRibbonToggleButton
	Telerik.Windows.Controls.Spreadsheet.dll	💾 RadRibbonView
	T-13. We-1 P-440	💷 RibbonButtonsPanel

现在做一个简单的示例,添加几个控件到界面,添加 AicVision/Misc 菜单下的 AlarmTextBlock 和 AnalogClock 控件。

MainWindow	8		
		123	
	o		c
	8		

4.2 数据绑定 DataBinding

数据绑定(Data Binding)是数据源与控件的关联,绑定的目标是控件中的某依赖项属性, 绑定的数据源是对象的公开属性。根据数据源和绑定方式的不同,数据绑定分为很多种。

4.2.1DataContext 绑定

在标记文件界面的 Grid 定义一个 DataContext, 然后切回设计界面, 点击 AlarmTextBlock

控件,在屏幕右下角的属性(Properties)窗口中找到此控件的 Value 属性,点击旁边的小方块,然后选择 Create Data Binding 建立数据绑定。

MainWindow.xaml	s × MainWindow.xaml* × Class1.cs ×	
🔧 MainWindow	•	•
4	mlnstaic="http://www.aic-tech.com/2012/xaml"	
6	Class= wprAppication2:mainwindow Title="MainWindow"	
7	Height="350"	
8	Width="525" DateContext="100">	
10 E <w< td=""><td>ndow.Resources></td><td></td></w<>	ndow.Resources>	
11 E <s< td=""><td>le x:Key="stl" TargetType="TextBlock"></td><td></td></s<>	le x:Key="stl" TargetType="TextBlock">	
12	<setter property="Text" value="ddddssst"></setter> 	
14 - </td <td>indow.Resources></td> <td></td>	indow.Resources>	
15 E <g< td=""><td>d DataContext="This is data context demo string."></td><td></td></g<>	d DataContext="This is data context demo string.">	
17	Height="119"	
18	Margin="72,34,0,0"	
19	VerticalAlignment="Top" Width="127" />	
21	<aic:alarmtextblock <="" horizontalalignment="Left" td=""><td></td></aic:alarmtextblock>	
22	Height="25"	
23	Margin="223,34,0,0" VerticelAlignment="Top"	
25	Width="196"	
26	Value="123" />	
27	< I EXTRIOCK X:INAME = "TEXTRIOCK" HorizontalAlianment = "Left"	
29	Height="20"	
30	Margin="223,133,0,0"	
31	TextWrapping="Wrap" VerticalAlignment="Top"	
33	Width="196"	
34	Style="{Binding Mode=OneWay, Source={StaticResource st}}"	
35 - 0</td <td>/> id></td> <td></td>	/> id>	
37 - <td>W></td> <td></td>	W>	
38		
Xaml Design		
Propertie	5	р ж
Propertie	S	4 ×
Propertie Name <	s No Name>	۹ × ها
Propertie Name < Type A	S No Name> armTextBlock	4 ×
Propertie Name < Type A	S No Name> armTextBlock	4 ×
Propertie Name < Type A	S No Name> armTextBlock	+ ×
Propertie Name < Type A	S No Name> larmTextBlock Style □ System.Windows.Style	4 × 4 © ©
Propertie Name < Type A	S No Name> armTextBlock Style = System.Windows.Style	× + 8 9 2
Propertie Name < Type A 24	S No Name> larmTextBlock Style = System.Windows.Style Template = System.Windows.Controls.ControlTemplate	× + • • •
Propertie Name < Type A 1	S No Name> armTextBlock Style System.Windows.Style Template System.Windows.Controls.ControlTemplate Alignment Center	+ × • • •
Propertie Name < Type A 24 Tex	S No Name> armTextBlock Style System.Windows.Style Template System.Windows.Controls.ControlTemplate Alignment Center	
Propertie Name < Type A 2 24 Tex	S No Name> armTextBlock Style = System.Windows.Style Template = System.Windows.Controls.ControlTemplate Alignment = Center Uid =	* * © © *
Propertie Name < Type A 2 21 Tex	S No Name> armTextBlock Style = System.Windows.Style Template = System.Windows.Controls.ControlTemplate Alignment = Center Uid =	
Propertie Name < Type A 21 21 Tex	S No Name> armTextBlock Style = System.Windows.Style Template = System.Windows.Controls.ControlTemplate Alignment = Center Uid = Value = 123	
Propertie Name Type A 21 Tex	S No Name> armTextBlock Style System.Windows.Style Template System.Windows.Controls.ControlTemplate Alignment Center Uid Curves Value 123	
Propertie Name Type A 24 Tex Securit	S No Name> armTextBlock Style System.Windows.Style Template System.Windows.Controls.ControlTemplate Alignment Center Uid Cuter Uid Local	
Propertie Name Type A 2 21 Tex Authorizat	S No Name> armTextBlock Style System.Windows.Style Template System.Windows.Controls.ControlTemplate Alignment Center Uid Center Uid 123 Value 123 Local on.Requi □	
Propertie Name Type A 2 24 Tex Authorizat	S No Name> armTextBlock Style = System.Windows.Style Template = System.Windows.Controls.ControlTemplate Alignment = Center Uid = Value = 123 Value = 123 Value = 123 Value = 123	
Propertie Name Type A 2 24 Tex Authorizat	S No Name> armTextBlock Style System.Windows.Style Template System.Windows.Controls.ControlTemplate Alignment Center Uid Center Uid 123 Value 123 Value 123 Value 123 Value 123	
Propertie Name Type A 2 2 2 Tex Authorizat Authorizat Authorizat	S No Name> armTextBlock Style System.Windows.Style Template System.Windows.Controls.ControlTemplate Alignment Center Uid Center Uid 123 Value 123 Value 123 Value 2000.Requi 0 con.Requi 0 con.Requ	
Propertie Name Type A 2 24 Tex Securit Authorizat Authorizat Authorizat	S No Name> armTextBlock Style System.Windows.Style Template System.Windows.Controls.ControlTemplate Alignment Center Uid Center Uid 123 Value 123 Local on.Requi Cocal on.Requi Cocal	
Propertie Name Type A 2 24 Tex Authorizat Authorizat Authorizat Authorizat	S No Name> armTextBlock Style System.Windows.Style Template System.Windows.Controls.ControlTemplate Alignment Center Uid Center Uid Center Uid Local on.Requi Cocal On.Req	
Propertie Name Type Authorizat Authorizat Authorizat Authorizat	S No Name> armTextBlock Style System.Windows.Style Template System.Windows.Controls.ControlTemplate Alignment Center Uid Center Uid Local on.Requi Cocal	

Custom Expression	
Reset	
Convert to Local Value	
Local Resource	
System Resource	Resources X
Edit Resource	
Convert to New Resource	
	lock
Create Data Binding	
Bind to Element	System.Windows.Style
Template Binding	System.Windows.Controls.ControlTemplate
Record Current Value	Center
Go to Source	
	= [100

在Binding Type中选择Data Context项,选择在Xaml中定义好的string类型的DataContext, 然后点击确定,将此DataContext 字符串绑定到AlarmTextblock的Value 属性上,在 MainWindow的Design界面可以看到AlarmTextblock显示了Xaml界面Grid中定义的DataContext的字符串值。

eate Data Binding for [AlarmTextBlock].Value		– O ×
nding Type: Data context		v
he data context is of type System.String.	Properties: String	Only display matching types
MainWindow	This is data context dem	no string
	dddssst	

4.2.2 StaticResource 绑定

我们做一个示例,在 Xaml 界面定义一个静态资源 Style, 键为"stl", 目标类型为 TextBlock.



为将此定义好的 StaticResource 绑定到我们的 TextBlock 控件中,我们在 TextBlock 控件的 Properties 窗口找到 Style 属性,点击旁边的小方块,选择 Creat DataBinding 建立数据绑定。

Properties		4 ×
Name textBlock		-
Type TextBlock		4
E 21		2
InputScope		-
IsHyphenationEnabl		
IsManipulationEnab		
OverridesDefaultStyle		
RenderTransformOr	0 0	
Resources	Collection)	
Style		
TextEffects	Collection)	
Uid		
 Security 		
Authorization.Requi		
Authorization.Requi		
· · · · - ·	- 1	

在 Binding Type 中选择 Explicit Source(StaticResource),选择已经定义的 stl, 点击 OK, 使 stl 这个 Style 绑定到 TextBlock 中。



绑定完成后 Xaml 中的 TextBlock 代码, TextBlock 中如果设置了 Text 属性,记得要删掉,不 然会因为优先级比 Style 高,而不显示静态资源 Style 中定义的 Text 值。



4.2.3 Element Name

Elemen Name 是对于绑定的 Source 来说的,指明绑定的源控件的名称。下面做一个示例,增加一个 TextBox 到界面

	This is data context demo string.	
Anseglieck		
	ddddssst	
	TextBox	

在 TextBox 的 Properties 属性框中找到 Text 属性,点击小方块,选择 Create Data Binding

Custom Expression		
Reset		
Convert to Local Value		
Local Resource		
System Resource	Resources ×	
Edit Resource		ቀ :
Convert to New Resource		
🔁 Create Data Binding		
Bind to Element		7
Template Binding		
Record Current Value	□ (2147483647	
Go to Source		
	TextBox	

在 BindingType 中选择 ElementName.

AicStudio 用户手册

Create Data Bi	nding for textBox.Text	
Binding Type:	ElementName	
Element name	Data context	` ———
clement name	ElementName	
 [Window] 	RelativeSource FindAncestor	
⊿ [Grid]	RelativeSource PreviousData	
[A	RelativeSource Self	
[A	RelativeSource TemplatedParent	
te	Explicit Source(StaticResource)	
te	OPC UA Data Node	
	OPC UA Relative Node 🗸	

然后选择 TextBlock 的 Text 属性作为绑定源。

reate Data Binding for textBox.Text		- 🗆
inding Type: ElementName		
Element name	Properties:	Only display matching type
▲ [Window]	Resources : (DictionaryEntry)	
▲ [Grid]	SnapsToDevicePixels : (Boolean)	
[AnalogClock]	Style : (Style)	
[AlarmTextBlock]	Tag : (Object)	
textBlock	TemplatedParent : (DependencyObject)	
textBox	Text : (String)	
	TextAlignment : (TextAlignment)	
	TextDecorations : (TextDecoration)	
	TextEffects : (TextEffect)	
	TextTrimming : (TextTrimming)	
	TextWrapping : (TextWrapping)	
	ToolTip : (Object)	
	TouchesCaptured : (IEnumerable < TouchD	evice>)
	TouchesCapturedWithin : (IEnumerable < T	ouchDevice>)
	TouchesDirectlyOver : (IEnumerable <touc< td=""><td>:hDevice>)</td></touc<>	:hDevice>)
	TouchesOver : (IEnumerable < TouchDevice	e>)
	Triggers : (TriggerBase)	
	Typography : (Typography)	
	Uid : (String)	
	UpdateDefaultStyle : (Method)	
	UpdateLayout : (Method)	
	UseLayoutRounding : (Boolean)	
	VerifyAccess : (Method)	
	VerticalAlignment : (VerticalAlignment)	
	Visibility : (Visibility)	
	Width : (Double)	
	Path: Text	
More settings		
		OK Consel
		UK Cancel

点击 OK,后可以看到 TextBox 显示出了绑定的字符串,在 Xaml 中可以看到 TextBox 进行绑定的代码。



4.2.4 OPC UA Data Node

OPC UA 数据节点绑定,是特殊类型的绑定,它的绑定的对象是标准的 OPC UA 服务器(例 如 AicDataHub) 中已经定义好的变量。

首先我们连接 DataHub,以本机的 DataHub 为例,先打开 DataHub 服务。

0.服务						操作	
AicTech DataHub Server	名称	描述	状态	启动类型	*	服务	-
	🔍 ActiveX Installer (AxInstSV)	为从		手动		更多操作	►
<u>停止</u> 此服务 重启动此服务	🔍 Adaptive Brightness	监视		手动	=	AicTech DataHub Server	
	🏟 AicTech DataHub Server	Data	已启动	手动		面	•
	🔍 AicTech IOServer	IOSe	已启动	手动		SCSP3#FF	
描述:	🔍 Application Experience	在应	已启动	手动			
DataHub server based on OPC	Application Identity	确定		手动			

然后在 AicStudio 中的 View 菜单下打开 OPC UA Server Explorer, 右键添加 Add New Connection

	Home	e Projec	t View	То	ols					
			D,	ı İQ	*	*		~*^		
Error	Output	Properties	Solution	Find	Tools	Outline	StartPage	Resources	OPC UA	
List			Explorer	Results					Server Explorer	
		C	Common					Miscellane	ous	



填写连接名称 Connection Name, Server Uri 地址,填写用户名 sa 和密码 sa (AicDataHub 默 认的用户名和密码,可以通过 AicDataHub Configurator 工具进行修改)。点击 Test 测试连接,出现 Connected OK 表示连接成功,点击 OK 即已经添加 OPC UA Connection。双击连接的名称,即可进行连接。

PC UA Server Information			
Connection Name	DataHub		
Description			
Server Uri	opc.tcp://127.0.0.1:4520		
Security Mode	None	▼ Tes	st
Security Policy	None	-	
Publishing Interval	1,000	‡ ms	
Session Timeout	600,000	‡ ms	
KeepAlive Interval	Connected OK. 5,000	‡ ms	
KeepAlive Error Threshold	3	* *	
Use Binary Encoding			
Authentication Settings — O Anonymous			
User Name	sa Password ••		
O Certificate	<bad certificate=""></bad>		
🔿 Issued Token			

连接完成后,我们可以进行 OPC UA Data Node 的绑定添加。

在上个示例的 TextBox 的 Properties 属性界面中找到 Text 属性,点击小方块,选择 Create Data Binding,在弹出的对话框中,Binding Type 选框中选择 OPC UA Data Node, Connection Name 选择 DataHub,点击 Refresh,获取此 OPC UA 的连接,若连接没问题的话,会出现此连接下的菜单项,Root/Objects 等。我们选择将 Text 属性绑定到 Root/Objects/Server/Auditing 下 LocalServerTimestamp.Data.Now,即当前本地服务时间。点击 OK

Connection Name: DataHub v Refresh New	Properties:	Only display matching typ
 Connection Name: Data Hub Data Hub Herresh Vew Connection Name: Data Hub Connection Name: Data Hub Connection Con	 Properues: UaVariable AccessLevel : (Byte) ArrayDimensions : (UInt32) BrowseName : (QualifiedName) DataType : (NodeId) DataValue : (DataValue) Description : (LocalizedText) DisplayName : (LocalizedText) DisplayName : (LocalizedText) DisplayName : (LocalizedText) DisplayName : (Boolean) IsGood : (Boolean) IsGood : (Boolean) IsValid : (Boolean) LocalServerTimestamp : (DateTime) Date : (DateTime) Date : (DateTime) Date : (DateTime) DayOfWeek : (DayOfWeek) DayOfWear : (Int32) Hour : (Int32) Kind : (DateTimeKind) Millisecond : (Int32) Month : (Int32) Month : (Int32) 	Only display matching y
	▲ Now : (DateTime) ► Date : (DateTime) Day : (Int32)	

在 MainWindow 界面,TextBox 显示了当前时间的字符串形式。

MainWindow	8	
	This is data context demo string.	
	ddddssst o	G
	2014-11-28T15:34:45.1921307	
	9	

4.2.5 OPC UA Relative Node

使用 OPC UA 节点相对绑定,可以动态切换绑定的对象,达到用一个显示界面可以切换 显示多个对象的效果,这是 AicVision 套件(AicStudio+AicDataHub+AicIOServer)提供的独一

无二的面向对象/设备组态的功能。

下面是一个简单的示例。首先添加显示的控件和按钮,两个按钮"站 101A"和"站 102" 用于在点击时切换绑定对象;标签(一次供水压力)旁边的 TextBox 表示"站 101A"或者"站 102"的一次供水压力的值。

MainWindow	
	This is data context demo string.
to z − p AnsingClock 4	ddddssst
	2014-11-28T17:28:55.4429426
站101A 站1	02
一次供水压力	

然后在系统的管理工具,服务中,打开 AicTech DataHub Server 服务。

◎ 服务				操作
AicTech DataHub Server	名称	描述 状态	启动类型 🔺	服务
	🖗 ActiveX Installer (AxInstSV)	为从	手动	更多操
<u>停止</u> 此服务 重户动业服务	🎑 Adaptive Brightness	监视	手动	AicTech Da
	端 AicTech DataHub Server	Data 已启动	手动	百名品
	AicTech IOServer	IOSe	手动	SC5/1#

打开 AicTech DataHub Configurator 进行连接(密码 sa)。

登录		-	- 1	3	х
服务器均	地址	opc.tcp://127.0.0.1:4520			
安全権	莫式	None 🔻			
安全領	策略	None 🔻			
用戶	⊐名	sa			
8	密码	••			
		确定即	消		

以下是以 HeatSupplyStation 为示例,用户可新建对象类型及其变量,并在工厂对象中添加新 建类型的对象。更多详细操作请参见 AicDataHub 文档资料。

			^
			-
	C)	复制	Ctrl+C
▲ I HeatSupp	Ж	剪切	Ctrl+X
- 🍕 二次供	<u>r</u>	粘贴	Ctrl+V
— 🔧 —次供	2	刷新	
— 🚰 补水泵	<u>.</u>	新建对象	类型
一 🚰 补水泵	UA1.	14/22	
- 🚰 补水泵	2频函	区馈	

然后打开 UaExpert,为 DataHub 中相关变量赋值。

T	De	efault DA View								
1	#	Server	Node Id	Display Name	Value	Datatype	ource Timestam	erver Timestam	Statuscode	
	1	AicTech DataHub Server AicTech DataHub Server	NS3 Numeric 103020 NS3 Numeric 103904	一次供水压力 一次供水压力	2 3	Float Float	9:59:42.487 9:59:44.168	9:59:42.523 9:59:44.168	Good Good	

开始进行相对绑定:

步骤一、父类容器控件中指定 BindingContext 节点的对象类型

在 Grid 中设置好 Name 属性,将 Grid 命名。

</window.kesources>
 Grid x:Name="BindingRoot"
 DataContext="This is data context demo string."

随意点击一个属性旁边的小方块,选择 Create DataBinding 进入数据绑定界面,在 BindingType 栏选择 OPC UA Relative Node 项,然后点击 Set BindingContext。

Create Data Binding for BindingRoot.Opacity			
Binding Type: OPC UA Relative Node			
Connection Name: DataHub	Refresh	Set BindingContext	Properties:

在 Set BindingContext 界面将命名 BindingRoot 的 Grid 与 DataHub 中用户定义的 HeatSupplyStation 类型进行绑定。注意: HeatSupplyStation 类型和此类型的对象"站 101" 等还有此类型下的"一次供水压力",都是 DataHub 工程中用户自定义的类型,如果本机 DataHub 没有此数据类型,可以登陆 DataHub 进行自定义,详细操作请参见 AicDataHub 文 档。

Target Element:	Set As Design Time Binding Context	Connection Name:	DataHub 🔻	Refresh	Ne
▲ [Window]		🔺 🗁 Root			
BindingRoot		🔺 🗁 Object	s		
[AnalogClock]		🕨 🔶 Ser	rver		
[AlarmTextBlock]		🕨 📄 Sys	stem		
textBlock		🔺 🗁 Types			
textBox		🕨 📄 Dat	taTypes		
button		🕨 🚞 Eve	entTypes		
button1		🔺 🗁 Ob	jectTypes		
label		📔 🔺 💐	BaseObjectType		
textBox2		►	🗟 AggregateConfigurationType		
			🗟 AggregateFunctionType		
		►	BaseConditionClassType		
		►	🗠 BaseEventType		
		⊿	🗠 CustomBaseObjectType		
			HeatSupplyStation		
			🗠 DataTypeEncodingType		
			🗠 DataTypeSystemType		
		▶	🗠 FileType		
			🗠 FolderType		
		►	🗠 Historical DataConfiguration Type		
		►	🗠 HistoryServerCapabilitiesType		
		►	🗠 LockType		
		▶	🗠 ModellingRuleType		
		►	🗠 NamespaceMetadataType		
		► ►	MamespacesType		

点击 OK,步骤一进行完毕。

步骤二、确定要显示的属性值

在要显示一次供水压力值的 TextBox 的 Properties 窗口中找到 Text 属性,点击旁边的小方块,选择 Create Data Binding 建立数据绑定,选择一次供水压力的 Value 属性。



步骤三、建立 DataHub 中实际对象与 BindingContext 的关联





在.xaml.cs 中实现 Click 方法



在站 101A 按钮的方法中的空白处点击右键,选择 Set Relative Binding Context,用脚本编辑器 自动添加 BindingContext 与实际对象关联的代码。

{ // TODO): Add	event handler implem	entation he	re.	NULLEUL VEITIMI YS EJ	
	₩	Cut	Ctrl+X			
		Сору	Ctrl+C			
1	1	Paste	Ctrl+V			
private void bu	\mathbf{x}	Delete	Del	ows.RoutedEventArgs e)		
{ // TODO	⊒	Comment/Uncomment Region		е.		
		Indent Selection				
		Rename	F2			
	-	Go To Definition	F12			
}		Insert Code Snippet	•		Set Relative Binding Context	
					Read variable's realtime data	
				*	Write variable's current value	
				*	Subscribe variable's realtime data	
				#	Read variable's history data	
				æ	Translate Path	
				*	Call Method	

点击 Refresh,按图中路径选择站 101A,点击 OK

Connection Name:	DataHub	*	Refresh	New
🔺 🗁 Root				
🔺 🗁 Objects				
🕨 🔧 Serve	r			
🔺 🗁 Syste	m			
► 🚞 A	larms			
► 🔩 G	lobalMethods			
🔺 🗁 P	lantObjects			
▲ <	\$ 太原			
	參訪101A			
•	⅔站101低区			
•	⅔站101高区			
•	⅔站102			
•	⅔站103中区			
•	⅔站103低区			
	43站103高区			
•	℃ 43 站105低区			
•	化站105高区			
	13 站107中区			
	13 站107低区			
P	13 5107新局			
Node Path: /Object	s/2:System/2:PlantObjects/2:太原/2:站101A		Node ID:	ns=3;i=1028
			ОК	Cancel



将 SetBindingContext 方法中的 this 参数改为 this.BindingRoot。 另外一个按钮的 Click 方法也按此步骤实现。最终代码如下图

AicStudio 用户手册



然后运行 Start Debugging,点击按钮控件进行测试

MainWindow	
	This is data context demo string. ddddssst 2014-11-28T17:58:25.6451924
站101A	占102
一次供水压力 2	

4.2.6 相对源绑定 RelativeSource

相对源绑定有 4 种模式, PreviousData, TemplatedParent, Self, FindAncestor 下图截至 System.Windows.Data.RelativeSourceMode

namespace System.Windows.Data



Self 模式:

Self 模式将元素自身的属性作为绑定的数据源。

示例:将 TextBox 的 ActualHeight 属性作为数据源绑定到 Text 属性上,进行显示。在 Properties 窗口找到 Text 属性,点击小方块,选择 CrateDataBinding 进入数据绑定界面。

Create Data Binding for textBox1.Text		– 🗆 ×
Binding Type: RelativeSource Self		.
	Properties:	Only display matching types
Bind textBox1 to itself.	▲ TextBox	
	AcceptsReturn : (Boolean)	
	AcceptsTab : (Boolean)	
	ActualHeight : (Double)	
	ActualWidth : (Double)	
	AllowDrop : (Boolean)	
	AreAnyTouchesCaptured : (Boolean)	
	AreAnyTouchesCapturedWithin : (Boolean)	
	AreAnyTouchesDirectlyOver : (Boolean)	
	AreAnyTouchesOver : (Boolean)	
	AutoWordSelection : (Boolean)	
	Background : (Brush)	
	BeginChange : (Method)	
	BeginInit : (Method)	
	BindingGroup : (BindingGroup)	
	BorderBrush : (Brush)	
	BorderThickness : (Thickness)	
	BringIntoView : (Method)	
	CacheMode : (CacheMode)	
	CanRedo : (Boolean)	
	CanUndo : (Boolean)	
	CaretBrush : (Brush)	
	CaretIndex : (Int32)	
	CharacterCasing : (CharacterCasing)	
	Clear : (Method)	
	Clip : (Geometry)	*
	Path: ActualHeight	

More settings

OK Cancel

选择 ActualHeight,点击 OK 确定绑定,绑定后的 Xaml 代码如图。

```
<TextBox x:Name="textBox1"
        HorizontalAlignment="Left"
        Height="26"
        Margin="289,173,0,0"
        TextWrapping="Wrap"
        Text="{Binding ActualHeight, Mode=OneWay, RelativeSource={RelativeSource Self}}"
        VerticalAlignment="Top"
        Width="35" />
```

FindAncestor 模式:

此模式的标记扩展可以实现访问此元素的 n 级父类的所有公开属性,作为绑定的数据源。 示例:选择一个 TextBox,将此控件的容器控件 Grid 的 Name 作为数据源,绑定到 Text 属性 上。

在 Text 属性上选择 CreateDataBinding,进入绑定窗口, BindingType 选择 RelativeSource FindAncestor,在 Ancestor type and level 框中输入 Grid 搜索出 Grid 控件,在右边 Properties 选框中选择 Name 属性,作为数据源进行绑定。

Create Data Binding for textBox3.Text	- • •	<
Binding Type: RelativeSource FindAncestor		٣
Ancestor type and level	Properties: Only display matching type	;
Grid × Image: Control of Contr	IsStylusCaptureWithin : (Boolean) IsStylusDirectlyOver : (Boolean) IsStylusOver : (Boolean) IsStylusOver : (Boolean) IsVisible : (Boolean) IsVisible : (Boolean) Language : (XmlLanguage) LayoutTransform : (Transform) LogicalOrientationPublic : (Orientation) Margin : (Thickness) MaxHeight : (Double) MaxWidth : (Double) MinHeight : (Double) MinWidth : (Double) Name : (String) OnApplyTemplate : (Method) Opacity : (Double) Parent : (DependencyObject) ReleaseAllTouchCaptures : (Method) ReleaseStylusCapture : (Method) RenderTransformOrigin : (Point) RenderTransformOrigin : (Point) RenderTransformOrigin : (Point) Reources : (DictionaryEntry)	
Show all assemblies	Path: Name	
More settings	OK Cancel]

点击 OK,绑定成功后的 Xaml 代码如下

```
<TextBox x:Name="textBox3"
HorizontalAlignment="Left"
Height="26"
Margin="289,226,0,0"
TextWrapping="Wrap"
Text="{Binding Name, RelativeSource={RelativeSource FindAncestor, AncestorType={x:Type Grid}}}"
VerticalAlignment="Top"
Width="83" />
```

TemplatedParent 模式:

RelativeSource 使用 TemplatedParent 模式时,仅在控件模板(ControlTemplate)或者数 据模板(DataTemplate)下有效。不同的模板,将返回不同类型的绑定结果。例如,在一个 ListBox 数据模板(DataTemplate)中应用 RelativeSource 的 TemplatedParent 模式,则会返回 ContentPresenter 模板内容到对应数据模板中。TemplatedParent 模式可以帮助开发人员轻松 绑定模板中的属性值到目标对象属性。例如:
使用 TemplatedParent 定义 Button 样式

```
<Style x:Key="sstl"

TargetType="{x:Type Button}">

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type Button}">

<TextBlock Background="Red"

Text="{Binding ActualHeight, Mode=OneWay, RelativeSource={RelativeSource TemplatedParent}}"

</ControlTemplate>

</Setter.Value>

</Setter>
```

```
</Style>
```

使用此样式的按钮 XAML 代码

<Button x:Name="button2" Style="{StaticResource sstl}" HorizontalAlignment="Left" Height="34" Margin="72,257,0,0" VerticalAlignment="Top" Width="81" />

按钮效果图,显示按钮高度。



值得注意的是,在控件模板(ControlTemplate)中使用 RelativeSource 的 TemplatedParent 模式,"Binding RelativeSource={RelativeSource TemplatedParent}}"等价于"{TemplateBinding}" 标记扩展。

两者不同在于,TemplateBinding 仅支持单向(One-Way)绑定,而 RelativeSource 标记 扩展支持双向(Two-Way)绑定,这个功能在创建自定义控件模板时特别有用。

PreviousData 模式

该模式很少使用,需要了解的可以参考微软 MSDN 中的相关内容: https://msdn.microsoft.com/zh-cn/library/system.windows.data.relativesource.previousdata%28 v=vs.110%29.aspx

4.3 动画 Animation(行为)

4.3.1 旋转(RotateAction)

在 AicStudio 中新建一个 wpf 界面,从控件区拖入一个 slider 控件如下图:



选中后适当调整控件长宽高,并在属性窗口设置其最大值(Maximum)和最小值(minimum)为 360 和 0,如下图:

Properti	ies	¢	×
Name	slider		⇔
Туре	Slider	l	Ş
E 21			\$
	Israbstop 🗆 💌		
	LargeChange 🛛 🛛		
	Maximum a 360		=
	Minimum D		
	Orientation D Horizontal	~	
	SelectionEnd 0		-

再从控件区拖入一个 RadButton 控件,适当调整大小,将其 Content 属性改为 Rotate, 在大纲处选中,如下图:

Out	line	ů ×
<u>↑</u>	[Window]	🛃 🕑 🔒
4	□ [Window]	
	▲ III [Grid]	@ <mark>0</mark>
	₽ slider	@ 0
	R radButton	0 0

选中后在工具窗口的左窗口选择行为 (Behaviors),在右窗口双击 LinearRotateAction 为 此按钮控件添加旋转动画。添加进来后在大纲中选中 radButton 下的 LinearRotateAction,如 图:

Out	tline	ů ×
<u>↑</u>	[Window]	🗗 🕑 🔒
4	□ [Window]	
	▲ III [Grid]	
	⊄ slider	
	⊿ 🖓 radButton	@ <mark>0</mark>
	🕞 [LinearRotateAction]	

选中后看属性窗口这个动画的属性如下图:

Properti	es	t ×
Name	<no name=""></no>	B
Туре	LinearRotateAction	\$
E 24		٩
🗢 Anir	mation	
🗢 Anir	mation/Source	
🗢 Anir	mation/Target	
Con	nmon	
🗢 Layo	out	
Seci	urity	
🕞 Trig	ger	

这里讲几个红线框中常用的几个属性,在 Animation 中 CenterX 和 CenterY 用于确定旋转的中心点,举个例子 CenterX 和 CenterY 都填 0.5,并且在 UseRelativeCenter 打上勾,则旋转的中心点就是这个按钮控件的中心点。注意:若要自己设置中心点,须在 UseRelativeCenter 打上勾,不打勾则是默认的中心点,为 0 和 0,即按钮控件的左上角的顶点。在此设置 CenterX 和 CenterY 都为 0.5,在 UseRelativeCenter 打上勾。



在 Animation/Target 中(如下图) AngleMax 和 AngleMin 用于设置旋转角度的范围,在 此设置最大角度为 360,最小角度为 0, ClockWise 为动画时旋转方向(在下文会详细介绍)。

Animation/Target
AngleMax ■ 360
AngleMin □ 0
Clockwise 🗆 📝

在 Animation/Target 中(如下图)设置最大值为 360,最小值为 0。说明:这里的最大 值、最小值和之后要绑定的 Value 值最终反应目标控件的旋转动画上时是按照比例进行运算 的,举个例子,如果最大值是 100,最小值是 50,Value 值是 75,则按钮控件实际的角度为 (75-50)/(100-50)*(360-0)+0=180,这里的 360 和 0是 Animation/Target 中 AngleMax 和 AngleMin 的属性值。如图中的 Value 值是 NaN,表示 Not a Number,这里需要为 Value 值进行绑定, 具体绑定的方法见本文档 4.2 数据绑定章节中的第 3 部分,这里将 Value 值绑定到 slider 的 Value 属性。TriggerOnValueChanged 默认是打勾的。(在下文会详细介绍)

Animation/Source	e
EnsureRange	
MaxValue	■ 360
MinValue	□ (0
TriggerOnValueCh	
Value	□ (NaN

至此,可以运行当前的程序可以看到效果如下图,滑动 slider 控件滑块,则按钮控件会绕其中心店旋转:



在 Trigger (如下图) 中,有 EventName 属性,在这里,这个属性是就是 radButton 所拥 有的事件,我们现在已经可以实现 radButton 的角度随 slider 拉动滑块进行变动,若将 Animation/Target 中的 TriggerOnValueChanged 的勾去掉,并选择 EventName 为 Click,则我们 可以看到滑动 slider 的滑块, radButton 并不旋转, 需要点击 radButton (出触发 Click 事件) 之后 radButton 转到对应的角度。



Properties Resources

在 Animation/Target 中,有个 Clickwise 属性:这个属性是动画过程中目标旋转的方向, 默认为顺时针,即 Clickwise 打上勾。说明:我们现在已经实现的功能主要的应用情况是: 实时地监测某一个物体的二位角度姿势(绑定实时数据库中相应的数据),这只能说是实时 监控的动画表现,对于这种动画,Clickwise 是不起任何作用的。其实,和 Clickwise 属性关 系很密切的一个属性是 Animation 中的 Duration 属性,这个属性的赋值举例为: 00:00:03, 这个就是动画持续的时间,此为3秒,这里所说的动画其含义为:从一个角度到另一个角度 所用的时间需要 3 秒 (如果 Duration 属性是默认值 Automatic 的话,从一个角度到另一个角 度是跳变的,我们现在已经实现的效果就是这种跳变式的)。我们可以做个试验:将 linearRotateAction 属性中的 Value 属性的数据绑定去掉(重新对 Value 赋值按 Tab 即可),从 工具窗口的控件区拖入一个 Button,将其 Content 属性设置为 RotateTest,在属性窗口点击 其事件按钮(如下图),在 Click 右边输入 Button Click,双击添加单击路由事件处理方法, 在方法中输入图示代码,设置 TriggerOnValueChanged 属性打上勾,设置 Animation 中的 Duration 属性为 00:00:01, 设置 Clickwise 属性打上勾,运行程序后点击后添加的按钮,动画 效果点击一次是顺时针转到 90 的位置, 若把 Clickwise 属性的勾去掉, 则是逆时针旋转到 90 度的位置。



private void Button_Click(object sender, System.Windows.RoutedEventArgs e)
{
 this.linearRotateAction.Value+=90;
}

4.3.2 缩放(ScaleAction)

从工具窗口的控件区再拖入一个 RadButton,将其 Content 属性设置为 Scale,按 4.3.1 中添加动画的方法,添加 LinearScaleAction 动画,此动画的属性中的 TriggerOnValueChanged 属性、EventName 属性和上文一样,不再赘述; Animation 中的 CenterX 和 CenterY 属性和旋转中的定义相同,(0,0)表示按钮的左上角的顶点,设置 CenterX 和 CenterY 为 0.5,在 UseRelativeCenter 属性打上勾; Animation/Source 中的 MinValue 和 MaxValue 属性设置为 0 和 360,同 4.3.1 旋转中的 Value 属性,将 Value 属性绑定到 slider 控件的 Value 属性;在 Animation/Target 中(如下图),缩放可以是只缩放 X 方向(横向)、只缩放 Y 方向(纵向),也可以是两个方向同时缩放,将 ScaleXEnabled 和 ScaleYEnabled 属性都打上勾,设置 ScaleXMax 为 0.5, ScaleXMin 为 1, ScaleYMax 为 2, ScaleMin 为 1,这里说明一点,不一定 Max 值一定要大于 Min,这几个值只是表示 X 方向和 Y 方向两个极端(最大和最小)的两种状态,联合 Animation/Source 中的 MaxValue、MinValue 和 Value,根据比例算的具体的缩放 状态,运行滑动 slider 滑块效果如图。



Rotate RotateTest	MainWindow	- 🗆 🗙	
	Rotate RotateTest	Scalekction	

4.3.3 尺寸变化(SizeAction)

从工具窗口的控件区再拖入一个 RadButton,将其 Content 属性设置为 SizeAction,按 4.3.1 中添加动画的方法,添加 LinearSizeAction 动画,此动画的属性中的 TriggerOnValueChanged 属性、EventName 属性、Duration 属性和上文一样,不再赘述。

此动画和 4.3.2 缩放较相似:都可以控制控件外形尺寸上的变化;都可以单独控制横向 或纵向的尺寸变化,也可以同时控制。区别在于:此动画没有 CnterX, CenerY 属性,默认 为左上角的顶点;缩放的尺寸变化是相对于原来控件的宽和高来说的(即成比例的放大或缩 小),但此动画为绝对的尺寸变化(和控件初始状态的宽和高无关),设置 Animation/Target 中的 HeightEnabled 和 WidthEnabled 都打上勾,设置 HeightMax 为 30, HeightMin 为 50, WidthMax 为 150, WidthMin 为 100, Animation 中的 MaxValue 为 360, MinValue 为 0, Value 属性绑定到 slider 控件的 Value 属性,运行滑动 slider 滑块效果如下图。从图中我们还可以 发现:此动画值变化尺寸,并不变化控件中显示的文字的大小,但 4.3.2 缩放动画不但改变 控件尺寸,也改变控件中的文字的尺寸。

	MainWindow	-	×
Rotate	RotateTest	ScaleAction	
SizeActio	n		

4.3.4 倾斜扭曲(SkewAction)

从工具窗口的控件区再拖入一个 RadButton,将其 Content 属性设置为 SkewAction,按 4.3.1 中添加动画的方法,添加 LinearSkewAction 动画,此动画的属性中的 TriggerOnValueChanged 属性、EventName 属性、Duration 属性和上文一样,不再赘述。

此动画的 CenterX 属性和 CenterY 属性和 4.3.1 旋转的对应属性相同,(0,0)表示控件左上角的顶点。设置 Animation/Source 的 MaxValue 属性为 360, MinValue 为 0, Value 属性绑定 到 slider 控件的 Value 属性; Animation/Target 中,设置 SkewXEnabled 和 SkewYEnabled 打上 勾,设置 SkewXMax 为 180, SkewXMin 为 0, SkwYMax 为 180, SkewYMin 为 0, 运行,滑动 slider 控件的滑块,效果如下图。



插入一个 Button 控件将 SkewAction 控件的 LinearSkewAction 的 Name 属性设置为 linearSkewAction1,将其 Content 属性设置为 SkewTest,删除 SkewAction 控件的 Value 的数 据绑定,按 4.3.1 添加 SkewTest 控件的单击事件,在添加的路由事件处理方法中添加如下代 码,设置 Duration 为 00:00:01,运行,点击 SkewTest 按钮,运行效果如下图。

□ priva □ { // th	nte void SkewTest_Cl TODO: Add event ha is.linearSkewAction1.	ick (object sender, System. Indler implementation here Value+=45;	Windows.RoutedEventA e.	٩rgs
		MainWindow	- • ×	
	Rotate	RotateTest	SkewTest	

4.3.5 平移(TranslateAction)

从工具窗口的控件区再拖入一个 RadButton,将其 Content 属性设置为 TranslateAction,按 4.3.1 中添加动画的方法,添加 LinearTranslateAction 动画,此动画的属性中的 TriggerOnValueChanged 属性、EventName 属性、Duration 属性和上文一样,不再赘述。

设置 Animation/Source 的 MaxValue 属性为 360, MinValue 为 0, Value 属性绑定到 slider 控件的 Value 属性; Animation/Target 中, Animation/Target 中, 设置 TranslateXEnabled 和 TranslateYEnabled 打上勾,设置 TranslateXMax 为 180, TranslateXMin 为 0, TranslateYMax 为 180, TranslateYMin 为 0, 运行, 滑动 slider 控件的滑块,效果如下图。



4.4 特效 Effects(效果)

AicStudio 一共集成了 17 种特效,它们都来自于 Microsoft.Expression.Media.Effects 命名 空间。所有的这些特效(包括模糊、锐化、滤镜、浮雕等)处理之后和效果和一些图像处理 软件(Photoshop)类似。在程序中使用特效与直接使用 PS 等软件进行图像处理相比,优势 在于这些特效的属性可以根据程序的需要非常方便的进行调整和改善,应用于动画中甚至可 以动态改变做出更炫目的效果。

AicStudio 用户手册

浏览:	自定义组件集	*	 G
<搜索	>		
	Microsoft.Expression.Effects		
) Microsoft.Expression.Media.Effects		
	▷ 🗗 BlindOrientation		
	👂 🔩 BlindsTransitionEffect		
	👂 🔩 BloomEffect		
	> 🔩 CircleRevealTransitionEffect		
	🖻 🔩 CloudRevealTransitionEffect		
	👂 🔩 CloudyTransitionEffect		
	👂 🔩 ColorToneEffect		
	🖻 🔩 EmbossedEffect		
	🛿 🔩 FadeTransitionEffect		
	▷ 🔩 MagnifyEffect		
	🕨 🔩 MonochromeEffect		
	Arrow Strate Pixelate Effect		
	PixelateTransitionEffect		
	A RadialBlurTransitionEffect		
	AndomizedTransitionEffect		
	A RippleEffect		
	RippleTransitionEffect		
	A SharpenEffect		
	SlideDirection		
	SlideInTransitionEffect		
	SmoothSwirlGridTransitionEffect		
	SwirlEffect		
	• State WaveTransitionEffect		
	▷ 🖶 WipeDirection		
	MipeTransitionEffect		

以 WPF 为例,这些特效都是 BitmapEffect(位图特效),使用 BitmapEffect 对所有 Visual 对象进行位图特效处理(比如文本,按钮,图像,甚至矩形、画布等),它是基于像素级别的,而且是基于软件处理模式而非硬件加速的处理模式。

BitmapEffect 在控件渲染时起作用,当控件渲染时,该 Visual 对象转化成相应的 BitmapSource 并作为 BitmapEffect 的输入,显示为经过 BitmapEffect 处理后的效果。

由于软件处理模式的效率相对较低,在有大量 Visual 对象或在有大量动画时慎用,此时 你必须在效果与效率之间做一个相对的平衡。

在 AicStudio 中,添加特效的操作非常简单,将选中的特效左键按住拖放到控件上即可。 每个控件仅允许存在一个特效。

T具箱		θ×	≤ MainWindow.xaml* × M	ainWindow.xaml.cs ×	MainWindow.xaml* × MainWindow.xaml.cs × MainWindow.xaml.cs × Ma
搜索 控件		٩			
 ○ ○<th>Ar BloomEffect Ar BloomEffect Ar ChexsboardEffect Ar ChexsboardEffect Ar ChexsboardEffect Ar ContratAdjustEffect Ar DropShadowEffect Ar EntratAdjustEffect Ar EntrataBlueEffect Ar HorizontalBlueEffect</th><th></th><th></th><th>MainWindow</th><th></th>	Ar BloomEffect Ar BloomEffect Ar ChexsboardEffect Ar ChexsboardEffect Ar ChexsboardEffect Ar ContratAdjustEffect Ar DropShadowEffect Ar EntratAdjustEffect Ar EntrataBlueEffect Ar HorizontalBlueEffect			MainWindow	
符号 ▶ 位置	J* Investmetiet J* InverColorsEffect J* MagnifyEffect J* MonochromeEffect J* RippleEffect J* RippleEffect J* SharpenEffect				Button

修改特效参数步骤为,在大纲窗口左键选中已添加的特效 BlurEffect,然后在特效的属性窗口中进行相关参数的修改;也可以在 Xaml 代码中直接进行修改。

大約	纲	4 ×
Ť	[Window]	🗗 🕑 🔒
4	□ [Window]	
	▲ III [Grid]	@ 0
	▲ 🖓 button	@ O
	Jx BlurEffect	٢

属性		θ×
名称 < 无名称 >		-
类型 BlurEffect		4
24		P
▲ 公共		
KernelType	□ Gaussian	•
Radius	4	
RenderingBias	Performance	•
▲ 布局		
FrameworkElement	□ LeftToRight	•

在 AicStudio 中,我们可以对控件进行以下几种常见的位图特效处理:

(1) 虚化效果(BlurEffect)

我们来看下虚化效果的简单示例:

添加 3 个按钮,第一个按钮无虚化效果,通过修改虚化效果的属性 Radius,第二个和第 三个的虚化效果逐渐增强。

虚化效果的属性窗口为

属性			đ 🗙
名称	<无名称>		-
类型	BlurEffect		4
21			2
<u>م</u> ک	共		
	KernelType	🗆 Gaussian	•
	Radius	■ 4	
	RenderingBias	□ Performance	•
▲布	局		
Frame	workElement	□ LeftToRight	•

最终的虚化显示效果如下

MainWindo	W
	未设置虚化特效的按钮
	虚化特效的按钮
	遗化特效的按钮

(2) 泛光效果(BloomEffect)

泛光效果可以使相对明亮的物体进行发光,下面我们做一个简单的示例。

添加 3 个椭圆 Ellipse,将它们的 Fill 属性设置为淡蓝色 LightBlue,然后将 BloomEffect 拖 放到 3 个椭圆上,在 BloomEffect 的属性窗口上将 Threshold(阈值)分别设置为 0.9,0.5,0.1

大纲	ф Х
1 [Window]	a
▲ □ [Window]	
▲ III [Grid]	@ <mark>0</mark>
 □ [Ellipse] 	@ <mark>0</mark>
Jx BloomEffect	٩
 □ [Ellipse] 	@ <mark>0</mark>
Jx BloomEffect	٩
 □ [Ellipse] 	@ <mark>0</mark>
∕≈ BloomEffect	٩
∕≭ BloomEffect	٩

属性	- 9 ×
名称 <无名称>	-
类型 BloomEffect	4
B 21	P
▲ 布局	
FrameworkElement LeftToRight	•
▲ 杂项	
BaseIntensity 🗆 🚺	
BaseSaturation 🗆 🚺	
BloomIntensity 🗆 (1.25	
BloomSaturation D	
Threshold	

最终的效果如图所示,从左到右,由于阈值减小,泛光效果的亮度明显增加。



(3) 阴影效果(DropShadowEffect)

阴影效果的示例也通过 3 个按钮来展示,通过改变各自的阴影效果的属性,达到合适的 阴影效果的目的。

大纲窗口,显示了在 button1 和 button2 下,各自拖放进去了一个 DropShadowEffect。

入羽	+	~
1 [Window]		_
▲ □ [Window]		
▲ III [Grid]	۲	0
다 button	٩	•
▲ 🖓 button1	٢	•
DropShadowEffect	0	
▲ ♀ button2	٢	•
/x DropShadowEffect	٩	

属性窗口显示了 DropShadowEffect 可以配置的属性。

属性		ф х
名称 <无名称 >		U
类型 DropShadowE	ffect	4
E 21		۶ ۲
▲ 公共		
BlurRadius	□ 5	
Color		
Direction	□ (315	
Opacity	□ 100%	
RenderingBias	Performance	•
ShadowDepth	5	
▲ 布局		
FlowDirection	□ LeftToRight	•

最终设置的阴影效果如图,其中第三个按钮的阴影效果将 Color (阴影颜色)属性设置为 Blue, ShadowDepth(阴影深度)设置为 10。

MainWindow	
	无阴影
	有阴影
	有明影

(4) 涡轮效果(SwirlEffect)

涡轮效果可以把平整的位图变为螺旋状的涡轮样式。

在 Xaml 界面放入 2 个按钮控件,第二个按钮控件添加涡轮效果(SwirlEffect)。Center 为涡轮效果的中心点,TwistAmount 属性为涡轮效果的扭曲程度。

属性				
名称	<无名称>			C
类型	SwirlEffect			(
E 21				
▲布				
		4To Platt		
	FlowDirection	ntrokignt		
(▲ 杂		TTOKIGHT		
▲ ♣	FlowDirection □ Let	5	0.5	

得到涡轮效果的对比图



(5) 浮雕特效(EmbossedEffect)

浮雕特效可以展现出雕刻的 3D 效果,现在我们做一个简单的示例。

在 Xaml 文本标记界面添加 2 个艾克信控的 AnalogClock 时钟控件,在第二个时钟控件上 拖放 EmbossedEffect 效果。在 EmBossedEffect 属性窗口配置其属性。

属性	4 ×
名称 <无名称 >	
类型 EmbossedEffect	εşu (
24	م
▲ 布局	
FlowDirection	LeftToRight •
🔺 杂项	
Amount 🗆	3
Color	
Height 🗆	0.001

将 Height(雕刻深度)设置为 0.001, Amount 设置为 3, Color 选为黄色, 效果显示如下



(6) 棋盘效果(ChessboardEffect) 棋盘特效的示例通过 2 个按钮来进行展示。

MainWindow



第二个按钮的棋盘效果参数窗口设置如下

属性	- 4 ×
名称 <无名称>	B
类型 ChessboardEffect	4
5 24	٩
▲ 布局	
FlowDirection CLeftToRight	•
▲ 杂项	
SquareSizeX ■ 0.2	
SquareSizeY (0.2	

SquareSizeX 和 SquareSizeY 两个参数范围为 0-1,表示分割的小块的单位,两个参数都为 0.2 表示整个按钮被分割成 5*5 的小块。

(7) 锐化效果(SharpenEffect)

锐化效果与模糊效果相反,可以提高位图的清晰度,使图像特定区域的色彩更加鲜明。 我们来做一个简单的示例,添加 2 个 Image 控件, Source 属性都为同一个模糊图片的路径, 然后在其中的一个 Image 控件上添加锐化效果。

属性			Υ Y
名称	<无名称>		\$
类型	SharpenEffect		Ş
21			2
▲布	局		
	FlowDirection	□ LeftToRight	•
<u>へ</u> 条	项		
	Amount	■ <u>6</u>	\square
	InputSize	□ 800,600	
	Scale	□ <u>1</u>	

通过调整锐化效果属性参数,获取更加清晰的图像显示效果。



(8) 像素化滤镜(PixelateEffect)

像素化滤镜将位图进行像素化的分割,看起来就像由一个个的小块组成。

我们放入 2 个艾克信控的 Towers004 控件,在第二个控件上拖放 PixelateEffect 效果,设置参数如下图所示。

属性	4 ×
名称 <无名称>	S
类型 PixelateEffect	\$
B 24	٩
▲ 布局	
FlowDirection CLeftToRight	•
▲ 杂项	
Pixelation 0.8	

得到的效果如图

AicStudio 用户手册



	53.	
States and the second sec		
II AF		
		55
		. A 🖬
) - EE

(9) 波纹效果(RippleEffect)

波纹效果将 Visual 位图变换为波纹的图像,以下是简单的示例。

在 Xaml 界面放入 2 个艾克信控的 RacingGague(速率仪表)控件,第二个控件拖放 RippleEffect,然后配置 RippleEffect 参数,将 Frequency(波纹频率)设置为 100.

属性	φ×
名称 <无名称>	\$
类型 RippleEffect	\$
B 91	٩
▲ 布局	
FlowDirection 🗆 LeftToRight	•
▲ 杂项	
Center 0.5 0.5	
Frequency (100	
Magnitude 🗆 0.1	
Phase 10	

下面是应用波纹效果后, 控件的效果对比图



(10) 放大镜效果(MagnifyEffect)

下面来一个简单的示例演示放大镜的效果,放大镜放大的不是控件,而是控件的显示元素。

在 Xaml 界面添加 2 个 Button,其中一个拖放 MagnifyEffect,然后设置其参数。Center 属性为放大的中心点,InnerRadius 和 OuterRadius 分别为放大镜的内角和外角(放大镜内外 边缘厚度),Amount 为放大镜离放大物体的距离。

属性	4 X
名称 <无名称>	-
类型 MagnifyEffect	4
2 24	٩
▲ 布局	
FlowDirection LeftToRight	•
杂项 杂项	
Amount 🗆 0.5	
Center 0.5 0.5	
InnerRadius 🗆 0.2	
OuterRadius 0.4	

调整完毕后的显示效果图如下。



(11) 色调(ColorToneEffect)

色调指的是一幅画中画面色彩的总体倾向,是大的色彩效果,我们可以通过色调效果改 变控件的显示。

做一个演示示例,在 Xaml 界面中放入 2 个 AnalogClock 控件。其中一个拖放 ColorEffect 效果,并配置其属性,因其默认的为亮黄色的暖色调,我们用 ColorEfeect 将之改为冷色调。

属性	4 ×
名称 < 无名称 >	
类型 ColorToneEffect	W 3
B 21	م
▲ 布局	
FlowDirection D LeftToRight	•
▲ 杂项	
DarkColor 🔳 💌	
Desaturation 2	
ToneAmount	

显示效果如图



(12) 对比度(ContrastAdjustEffect)

对比度指的是一幅图像中明暗区域最亮的白和最暗的黑之间不同亮度层级的测量,差异 范围越大代表对比越大,差异范围越小代表对比越小。

对比度对视觉效果的影响非常关键,一般来说对比度越大,图像越清晰醒目,色彩也越 鲜明艳丽;而对比度小,则会让整个画面都灰蒙蒙的。



来做一个简单的示例,在 Xaml 界面中放入 2 个 AnalogClock 控件,其中一个拖放 ContrastAdjustEffect,然后配置其属性, Brightness 属性增加亮度, Contrast 属性配置对比度。

属性	4 ×
名称 <无名称>	\$
类型 ContrastAdjustEffect	\$
B 24	م
◆ 布局	
FlowDirection D	•
▲ 杂项	
Brightness D	
Contrast	

配置完毕的效果如图,可以看出,对比度提高后更加清晰。



(13)饱和度(SaturationEffect)饱和度是指色彩的鲜艳程度,也称色彩的纯度。



5 脚本

5.1 代码编辑器

在 AicStudio 打开类文件(.cs 文件)的同时,我们进入 C#代码编辑器界面,编辑器有强大的代码编辑相关功能,人性化的智能提示工具,自动添加代码的脚本向导等。



5.2 脚本向导

AicStudio 的脚本向导是为代码编辑服务,自动添加代码实现与 OPC UA Server 的交互, 省去了添加引用程序集和命名空间引用,寻找适合的方法等步骤,简化了代码编辑操作,提 高编辑效率。

在可编辑代码的位置,右键单击,选择 Insert Code Snippet 插入代码段,目前主要有 7 项代码段提供便捷插入。

var c 🐰	Cut	Ctrl+X	Incor	["DataHub"] as IllaConnectionCo
if (cc 🗎	Сору	Ctrl+C	Calid) {	{
, t 🛍	Paste	Ctrl+V	usCo	odes.BadConfigurationError);
· ×	Delete	Del		
UaBir t ⊒ r	Comment/Uncomment Region Indent Selection		binding target element(normally use w UaNodePath("/Objects/2:System/	
	Rename	F2		
-	Go To Definition	F12		
var c	Insert Code Snippet	1	> ant	Set Relative Binding Context
			4	Read variable's realtime data
Duild =			4	Write variable's current value
bulia +				Subscribe variable's realtime data
				Read variable's history data
				Translate Path

5.2.1 设置相对绑定对象(Set Relative Binding Context)

此项请参见 4.2 节数据绑定的第五部分,在设置 OPC UA 节点相对绑定的时候,就使用 过此脚本添加相应的代码段,将根节点绑定数据上下文,子节点的相对路径切换时,相对绑 定不同的对象随之切换。

5.2.2 修改变量当前值(Write variable's current value)

我们为此代码段做一个简单的示例,用来修改本机 DataHub 中的某一变量的值,为了 验证是否已经修改,我们将修改后的值进行显示。选定的变量可以根据自身本机 DataHub 中己有的进行选择,也可以新建变量(参考 AicDataHub 文档)。

步骤一、新建 WPF 工程 WriteCurrentValueDemo

ガ突		模板
⊿ 🜈 C#	:	멸WPF Application
Silverlight		聽WPF Control Library
4 project t	for creating rich desk WriteCurrentValueD	top applications that run on Windows.
A project 名称 路径	for creating rich desk WriteCurrentValueD C:\Users\dst\Docum	top applications that run on Windows. Demo nents\AicStudio\Projects\WpfApplication2\

步骤二、添加控件,这里添加了 Button,Label,TextBox,TextBlock。



步骤三、为按钮添加单击事件,添加修改变量值的代码段 在要添加代码段的空白处右键单击选择插入代码段->修改变量当前值。



这里的示例选择节点,选中 Float 类型的变量"二次供水压力"

选择节点				- 0
连接名称:	*DataHub	•	刷新	新建
🖌 🗁 Roo	t			
4 🧁	Objects			
►	🛠 Server			
4	🧁 System			
	Alarms			
	A GlobalMethods			
	PlantObjects			
	▲ ☆ 站101A □ □			
	☆ 补水泵1频率反馈			
	☆ 补水泵1运行状态			
	☆ 补水泵2频率反馈			
	11 补水泵2运行状态			
	☐ 次·			
5点路径:	/Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力		节点ID:	ns=3;i=10287
			确定	取消

点击确定后,可以看到已经在按钮事件处理方法中添加了修改变量值的代码,图中标示红色字体 TODO 的部分为可以自行修改的部分。

```
private void button_Click(object sender, System.Windows.RoutedEventArgs e)
       {
           // TODO: Add event handler implementation here.
           var configuration = Application.Current.Resources["DataHub"] as IUaConnectionConfiguration;
           if (configuration == null || !configuration.IsValid) {
               throw new ServiceResultException(StatusCodes.BadConfigurationError);
           var connection = UaConnectionManager.Current.GetConnection(configuration);
           var session = (connection == null) ? null : connection.Connection:
           if (session == null) {
               throw new ServiceResultException(StatusCodes.BadNotConnected);
           var valueToWrite = new WriteValue();
           valueToWrite.NodeId = new NodeId(102872U, 3); // /Objects/2:System/2:PlantObjects/2:法息/2:站101A/2:二次供水压力
           valueToWrite.AttributeId = Attributes.Value:
           valueToWrite.Value = new DataValue(StatusCodes.Good);
           valueToWrite.Value.Value = (Single)0.0; // TODO: replace with real variable value(System.Single)
           session.WriteAttributeAsync(valueToWrite).ContinueWith((t) => {
               if (t.Exception != null) {
                  // TODO: Error
              } else {
                  if (ServiceResult.IsBad(t.Result)) {
                      // TODO: Error
                  } else {
                      // TODO: OK
                  3
          }, TaskScheduler.FromCurrentSynchronizationContext());
       }
我们将修改的新值确定为 TextBox 中输入的值。
 private void button_Click(object sender, System.Windows.RoutedEventArgs e)
   {
       // TODO: Add event handler implementation here.
       var configuration = Application.Current.Resources["DataHub"] as IUaConnectionConfiguration;
       if (configuration == null || !configuration.IsValid) {
           throw new ServiceResultException(StatusCodes.BadConfigurationError);
       }
       var connection = UaConnectionManager.Current.GetConnection(configuration);
       var session = (connection == null) ? null : connection.Connection;
       if (session == null) {
           throw new ServiceResultException(StatusCodes.BadNotConnected);
       }
       var valueToWrite = new WriteValue();
       valueToWrite.NodeId = new NodeId(102872U, 3); // /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力
       valueToWrite.AttributeId = Attributes.Value;
       valueToWrite.Value = new DataValue(StatusCodes.Good);
       valueToWrite.Value.Value = Single.Parse(this.textBox.Text); // TODO: replace with real variable value(System.Single)
       session.WriteAttributeAsync(valueToWrite).ContinueWith((t) => {
           if (t.Exception != null) {
               // TODO: Error
           } else {
               if (ServiceResult.IsBad(t.Result)) {
                   // TODO: Error
               } else {
                   // TODO: OK
               }
       }, TaskScheduler.FromCurrentSynchronizationContext());
   }
```

然后将此变量的 Value 属性绑定到 TextBlock.Text 依赖项属性,将此变量的值在 TextBlock 显示。

为 textBlock.Text 创建数据绑定	– = ×
網定类型 OPC UA绝对绑定	·
连接名称: DataHub 新建	属性 口 反显示匹配类型
A 🗁 Root	▲ UaVariable
 Dijects 	AccessLevel : (Byte)
A Server	ArrayDimensions : (UInt32)
🔺 🧁 System	BrowseName : (QualifiedName)
Alarms	DataType : (NodeId)
Ag GlobalMethods	DataValue : (DataValue)
PlantObjects	Description : (LocalizedText)
▲ 餐太原	DisplayName : (LocalizedText)
▲ 😚站101A	Dispose : (方法)
🚰 补水泵1频率反馈	Historizing : (Boolean)
🚰 补水泵1运行状态	IsGood : (Boolean)
計补水泵2频率反馈	IsValid : (Boolean)
計补水泵2运行状态	LocalServerTimestamp : (DateTime)
計补水泵3频率反馈	LocalSourceTimestamp : (DateTime)
m 补水泵3运行状态	MinimumSamplingInterval : (Double)
▶ m 二次供水温度	NodeClass : (NodeClass)
▶ 13 二次供水温度报警	NodeId : (NodeId)
	NodePath : (UaNodePath)
 ** 二次供水压力报警 	ServerTimestamp : (DateTime)
	SourceTimestamp : (DateTime)
▶ 〒二次回水压力	StatusCode : (StatusCode)
	Tag : (Object)
	UserAccessLevel : (Byte)
	UserWriteMask : (UInt32)
	Value : (Object)
11111111111111111111111111111111111111	ValueRank : (Int32)
节点路径: /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力	網定路径 Value

🔻 更多设置

确定 取消

点击确定后,将此工程设置为启动工程,运行测试。

MainWindow		
修改变量值 变量值显示	3.678 3.678	

5.2.3 读取变量实时值(Read variable's realtime data)

读取变量实时值为一个取值的过程,这里的示例将值取出并进行显示。 此示例沿用了 2)中的全部代码,并进行进一步添加,增加读取变量值的按钮和显示读取结 果的 TextBlock.

MainWindow		
	6	
	修改 ²⁸ 量值 TextBox	
	变量值显示	
co — 102 —	c 读取交量值 TextBlock	•

读取变量值按钮添加单击事件,并右键添加读取变量实时值的代码段。 private void button1_Click(object sender, System.Windows.RoutedEventArgs e)



选择节点

	接名称:	DataHub	-	刷新	新建
<pre>* cont * ***********************************</pre>				AP3491	4/1/2
 ● Option ● System ● System ● PlantOyects /ul>		Dhierte			
<pre>variabled = new Nodeld(102872U, 3)// /Objects/2:5ystem/2PlantObjects {</pre>	- <u>-</u>	A Server			
<pre>> * *********************************</pre>		System			
<pre> * %GlobalMethods * @PlantOpiets * %globalMethods * @PlantOpiets * %glam * %globalMethods * %globalMet</pre>	_	Alarms			
<pre> PhantObjets</pre>	1	GlobalMethods			
<pre> the set of /pre>		PlantObjects			
<pre> *********************************</pre>		▲ ⁴ 3 太原			
<pre></pre>		▲ % 站101A			
<pre></pre>		🚰 补水泵1频率反馈			
<pre></pre>		🚰 补水泵1运行状态			
<pre>If http://www.setuple.com/setuple.c</pre>		🚰 补水泵2频率反馈			
■ **水果要要 ● **: 二次供水温要报警 ● **: 二次供水温要 ● **: 二次供水温要 ● **: 二次回水温 ● **: 二次 ● **: 三次 ● **: 二次 ● **: 二次 ● **: 二次 ● **: ● **: ● **: ● **: ● **: ● **: ● **: ● **: ● **: ● **: ● **: ● **: ● **: ● **: ● **: ● **: ● **: ● **: ● **: ● *: ● **: ● **: ● **: ● *		🚰 补水泵2运行状态			
■ **水系活動で ● **		🚰 补水泵3频率反馈			
<pre> * 管 二次供水温度报管 * *******************************</pre>		🚰 补水泵3运行状态			
<pre>> %t=二次供水压力 > %t=二次供水压力 = %t=二次回水温度 > 還一次回水压力 : 還力率 : 還一次回水压力 : 還力率 : 還一次回水压力 : 還力率 : 還一次回水压力 : 還力率 : 還加压泵15% : 還一次回水压力 : : ns=3j=1028 : : : : : : : : : : : : : : : : : : :</pre>		▶ 🚰 二次供水温度			
<pre>> (*) (*) (*) (*) (*) (*) (*) (*</pre>		▶ 🏫 二次供水温度报警			
<pre></pre>		▶ @ 二次供水压力			
<pre>>> ■ 二次回次混反 >> ■ 二次回次混反 @ 加压泵1本地运程 @ 加压泵1本地运程 @ 加压泵1本地运程 @ 加压泵1本地运程 @ 加压泵1本地运程 @ 加压泵14位 和定 D2 和定 D2 和C 和定 D2 和C 和C 和C 和C 和C 和C 和C 和C 和C 和C</pre>		▶ 🍕 二次供水压力报警			
<pre>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>></pre>					
<pre></pre>		▶ 1 二次回水压力			
Im Tut::::::::::::::::::::::::::::::::::::					
Implementation Implementati					
Implicite Table 点歸徑: /Objects/2:System/2:PlantObjects/2:太原/2:bi101A/2:二次供水压力 南定 取消 请確定 取消 許確定 「京加口: multiple: multiple		1117加压泉1反馈			
<pre>index = i volgetarely intervention interospectarely undex version versio</pre>					
<pre>impact = 1.5 # 1.5</pre>	占路径・	//Objects/2:System/2:PlantObjects/2:太原/2:訪101A/2·一次併水压力		파모ID ·	ns=3·i-1029
<pre>inde定 后添加的代码段如下 invate void button1_Click(object sender, System.Windows.RoutedEventArgs e) { // TODO: Add event handler implementation here. var configuration = Application.Current.Resources["DataHub"] as IUaConnectionConfiguration; if (configuration == null !configuration.ISValid) { throw new ServiceResultException(StatusCodes.BadConfigurationError); } var connection = UaConnectionManager.Current.GetConnection(configuration); var session = (connection=null) ? null : connection.Connection; if (session == null) ? throw new ServiceResultException(StatusCodes.BadNotConnected); } var variableId = new NodeId(102872U, 3);// /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力 session.ReadValueAsync(variableId).ContinueWith((t) => { if (t.Exception != null) { // TODO: Error } else { // TODO: OK // TaskScheduler.FromCurrentSynchronizationContext()); } </pre>	点路径:	灣加压泵1給定 /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力		节点ID:	ns=3;i=1028
<pre>ivate void button1_Click(object sender, System.Windows.RoutedEventArgs e) { // TODO: Add event handler implementation here. var configuration = Application.Current.Resources["DataHub"] as IUaConnectionConfiguration; if (configuration == null !configuration.IsValid) { throw new ServiceResultException(StatusCodes.BadConfigurationError); } var connection = UaConnectionManager.Current.GetConnection(configuration); var session = (connection== null) ? null : connection.Connection; if (session == null) { throw new ServiceResultException(StatusCodes.BadNotConnected); } var variableId = new NodeId(102872U, 3);// /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力 session.ReadValueAsync(variableId).ContinueWith((t) => { if (t.Exception != null) { // TODO: Error } else { // TODO: OK // TODO: Addet = t.Result } } // Tobo: OK // To</pre>	点路径:	────────────────────────────────────		节点ID: 确定	ns=3;i=1028 取消
<pre>{ // TODO: Add event handler implementation here. var configuration = Application.Current.Resources["DataHub"] as IUaConnectionConfiguration; if (configuration == null configuration.IsValid) { throw new ServiceResultException(StatusCodes.BadConfigurationError); var connection = UaConnectionManager.Current.GetConnection(configuration); var session = (connection == null) ? null : connection.Connection; if (session == null) { throw new ServiceResultException(StatusCodes.BadNotConnected); var variableId = new NodeId(102872U, 3);// /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力 session.ReadValueAsync(variableId).ContinueWith((t) => { if (t.Exception != null) { // TODO: Error } else { // TODO: OK // TODO: OK // TODO: CK // TODO: OK // TODO: CK // TODO: A // TODO: CK // TODO: CK</pre>		赠加压氡1给定 /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力 后添加的代码段如下		节点ID: 确定	ns=3;i=1028 取消
<pre>// TODO: Add event handler implementation here. var configuration = Application.Current.Resources["DataHub"] as IUaConnectionConfiguration; if (configuration == null !configuration.IsValid) { throw new ServiceResultException(StatusCodes.BadConfigurationError); } var connection = UaConnectionManager.Current.GetConnection(configuration); var session = (connection == null) ? null : connection.Connection; if (session == null) { throw new ServiceResultException(StatusCodes.BadNotConnected); } var variableId = new NodeId(102872U, 3);// /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力 session.ReadValueAsync(variableId).ContinueWith((t) => { if (t.Exception != null) { // TODO: Error } else { // TODO: OK // TODO: OK // Yar dataValue = t.Result; } }, TaskScheduler.FromCurrentSynchronizationContext());</pre>	点路径: 亍确定/	☞加压泵1给定 /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力 后添加的代码段如下 id button1 Click(object sender, System,Windows,RoutedEventArgs e)		节点ID: 确定	ns=3;i=1028 取消
<pre>var configuration = Application.Current.Resources["DataHub"] as IUaConnectionConfiguration; if (configuration == null !configuration.IsValid) { throw new ServiceResultException(StatusCodes.BadConfigurationError); } var connection = UaConnectionManager.Current.GetConnection(configuration); var session = (connection == null) ? null : connection.Connection; if (session == null) { throw new ServiceResultException(StatusCodes.BadNotConnected); } var variableId = new NodeId(102872U, 3);// /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力 session.ReadValueAsync(variableId).ContinueWith((t) => { if (tException != null) { // TODO: Error } else { // TODO: OK // TODO: OK // var dataValue = t.Result; } }, TaskScheduler.FromCurrentSynchronizationContext());</pre>	点路径: 亍确定/ rivate vo {	☞加压泵1给定 /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力 后添加的代码段如下 id button1_Click(object sender, System.Windows.RoutedEventArgs e)		节点ID: 确定	ns=3;i=1028 取消
<pre>if (configuration == null []:configuration.isvalid) { throw new ServiceResultException(StatusCodes.BadConfigurationError); } var connection = UaConnectionManager.Current.GetConnection(configuration); var session = (connection == null) ? null : connection.Connection; if (session == null) { throw new ServiceResultException(StatusCodes.BadNotConnected); } var variableId = new NodeId(102872U, 3);// /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力 session.ReadValueAsync(variableId).ContinueWith((t) => { if (t.Exception != null) { // TODO: Error } else { // TODO: OK // TODO: OK // TODO: OK // ToDO: OK // TaskScheduler.FromCurrentSynchronizationContext()); </pre>	点路径: 亍确定/ rivate vo { // 】	☞加压泵1给定 /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力 后添加的代码段如下 id button1_Click(object sender, System.Windows.RoutedEventArgs e) ODO: Add event handler implementation here.		节点ID: 确定	ns=3;i=1028 取消
<pre>kinow new ServiceResultException(dataSeodeS.BddComection(configuration); var session = (connection == null) ? null : connection.Connection; if (session == null) { throw new ServiceResultException(StatusCodes.BadNotConnected); } var variableId = new NodeId(102872U, 3);// /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力 session.ReadValueAsync(variableId).ContinueWith((t) => { if (t.Exception != null) { // TODO: Error } else { // TODO: OK // TODO: OK // TODO: OK // TaskScheduler.FromCurrentSynchronizationContext()); </pre>	点路径: 亍确定) rivate vo { // 1 var	☞加压泵1给定 /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力 后添加的代码段如下 id button1_Click(object sender, System.Windows.RoutedEventArgs e) ODO: Add event handler implementation here. configuration = Application.Current.Resources["DataHub"] as IUaConnection	onCon	节点ID: 确定 figuration;	ns=3;i=1028 取消
<pre>var connection = UaConnectionManager.Current.GetConnection(configuration); var session = (connection == null) ? null : connection.Connection; if (session == null) { throw new ServiceResultException(StatusCodes.BadNotConnected); } var variableId = new NodeId(102872U, 3);// /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力 session.ReadValueAsync(variableId).ContinueWith((t) => { if (t.Exception != null) { // TODO: Error } else { // TODO: OK [/ var dataValue = t.Result; } }, TaskScheduler.FromCurrentSynchronizationContext());</pre>	点路径: 亍确定) rivate vo { // 1 var if (☞加压泵1给定 /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力 后添加的代码段如下 id button1_Click(object sender, System.Windows.RoutedEventArgs e) ODO: Add event handler implementation here. configuration = Application.Current.Resources["DataHub"] as IUaConnectio configuration == null !configuration.IsValid) {	onCon	节点ID: 确定 figuration;	ns=3;i=1028 取消
<pre>var connection = UaConnectionManager.Current.GetConnection(configuration); var session = (connection == null) ? null : connection.Connection; if (session == null) { throw new ServiceResultException(StatusCodes.BadNotConnected); } var variableId = new NodeId(102872U, 3);// /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力; session.ReadValueAsync(variableId).ContinueWith((t) => { if (t.Exception != null) { // TODO: Error } else { // TODO: OK [/ var dataValue = t.Result; } }, TaskScheduler.FromCurrentSynchronizationContext());</pre>	点路径: 亍确定) rivate vo { // 1 var if (r	☞加压泵1给定 /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力 后添加的代码段如下 id button1_Click(object sender, System.Windows.RoutedEventArgs e) ODO: Add event handler implementation here. configuration = Application.Current.Resources["DataHub"] as IUaConnectio configuration == null !configuration.IsValid) { throw new ServiceResultException(StatusCodes.BadConfigurationError);	onCon	节点ID: 确定 figuration;	ns=3;i=1028 取消
<pre>var session = (connection == null) ? null : connection.Connection; if (session == null) { throw new ServiceResultException(StatusCodes.BadNotConnected); } var variableId = new NodeId(102872U, 3);// /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压大 session.ReadValueAsync(variableId).ContinueWith((t) => { if (t.Exception != null) { // TODO: Error } else { // TODO: OK [/ var dataValue = t.Result; } }, TaskScheduler.FromCurrentSynchronizationContext());</pre>	点路径: 〒确定) rivate vc { // 1 var if (☞加压泵1给定 /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力 后添加的代码段如下 id button1_Click(object sender, System.Windows.RoutedEventArgs e) ODO: Add event handler implementation here. configuration = Application.Current.Resources["DataHub"] as IUaConnectio configuration == null !configuration.IsValid) { throw new ServiceResultException(StatusCodes.BadConfigurationError);	onCon	节点ID: 确定 figuration;	ns=3;i=1028 取消
<pre>if (session == null) { throw new ServiceResultException(StatusCodes.BadNotConnected); } var variableId = new NodeId(102872U, 3);// /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力 session.ReadValueAsync(variableId).ContinueWith((t) => { if (t.Exception != null) { // TODO: Error } else { // TODO: OK // TODO: OK // var dataValue = t.Result; } }, TaskScheduler.FromCurrentSynchronizationContext()); </pre>	点路径: fi确定) fivate vc { // 1 var if (} var	☞加压泵1给定 /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力 后添加的代码段如下 id button1_Click(object sender, System.Windows.RoutedEventArgs e) ODO: Add event handler implementation here. configuration = Application.Current.Resources["DataHub"] as IUaConnection configuration == null !configuration.IsValid) { throw new ServiceResultException(StatusCodes.BadConfigurationError); connection = UaConnectionManager.Current.GetConnection(configuration	onCon	节点ID: 确定 figuration;	ns=3;i=1028 取消
<pre>} var variableId = new NodeId(102872U, 3);// /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力 session.ReadValueAsync(variableId).ContinueWith((t) => { if (t.Exception != null) { // TODO: Error } else { // TODO: OK (// var dataValue = t.Result; } }, TaskScheduler.FromCurrentSynchronizationContext());</pre>	点路径: 宁确定/ rivate vc { //1 var if (r } var var	☞加压泵1给定 /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力 后添加的代码段如下 id button1_Click(object sender, System.Windows.RoutedEventArgs e) ODO: Add event handler implementation here. configuration = Application.Current.Resources["DataHub"] as IUaConnectio configuration == null !configuration.IsValid) { throw new ServiceResultException(StatusCodes.BadConfigurationError); connection = UaConnectionManager.Current.GetConnection(configuration session = (connection == null) ? null : connection.Connection; terviceResultException(StatusCodes.BadConfiguration	onCon	节点ID: 确定 figuration;	ns=3;i=1028 取消
<pre>var variableId = new NodeId(102872U, 3);// /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力 session.ReadValueAsync(variableId).ContinueWith((t) => { if (t.Exception != null) { // TODO: Error } else { // TODO: OK (/ var dataValue = t.Result; } }, TaskScheduler.FromCurrentSynchronizationContext());</pre>	点路径: F确定/ fivate vc { ///1 var if (r var var var if (r	☞加压泵1给定 /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力 言添加的代码段如下 id button1_Click(object sender, System.Windows.RoutedEventArgs e) ODO: Add event handler implementation here. configuration = Application.Current.Resources["DataHub"] as IUaConnectio configuration == null !configuration.IsValid) { throw new ServiceResultException(StatusCodes.BadConfigurationError); connection = UaConnectionManager.Current.GetConnection(configuration session = (connection == null) ? null : connection.Connection; session == null) { throw new ServiceResultException(StatusCodes.BadNotConnected);	nCon	节点ID: 确定 figuration;	ns=3;i=1028 取消
<pre>var variableId = new NodeId(102872U, 3);// /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力 session.ReadValueAsync(variableId).ContinueWith((t) => { if (t.Exception != null) { // TODO: Error } else { // TODO: OK // TODO: OK // var dataValue = t.Result; } }, TaskScheduler.FromCurrentSynchronizationContext()); </pre>	点路径: 亍确定丿 rivate vc { // 1 var if (! var if (! }	☞加压泵1给定 /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力 言添加的代码段如下 id button1_Click(object sender, System.Windows.RoutedEventArgs e) ODO: Add event handler implementation here. configuration = Application.Current.Resources["DataHub"] as IUaConnection configuration == null !configuration.IsValid) { throw new ServiceResultException(StatusCodes.BadConfigurationError); connection == null) ? null : connection.Connection; session == null) {	nCon	节点ID: 确定 figuration;	ns=3;i=1028 取消
<pre>session.ReadValueAsync(variableId).ContinueWith((t) => { if (t.Exception != null) { // TODO: Error } else { // TODO: OK (/ var dataValue = t.Result; } }, TaskScheduler.FromCurrentSynchronizationContext());</pre>	点路径: F确定丿 rivate vc { // 1 var if (} var if (; }	☞加压泵1给定 /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力 合添加的代码段如下 id button1_Click(object sender, System.Windows.RoutedEventArgs e) ODO: Add event handler implementation here. configuration = Application.Current.Resources["DataHub"] as IUaConnection configuration == null !configuration.IsValid) { throw new ServiceResultException(StatusCodes.BadConfigurationError); connection = UaConnectionManager.Current.GetConnection(configuration session = (connection == null) ? null : connection.Connection; session == null) { throw new ServiceResultException(StatusCodes.BadNotConnected);	nCon	节点ID: 确定 figuration;	ns=3;i=1028 取消
<pre>ir (LException != null) { // TODO: Error } else { // TODO: OK // var dataValue = t.Result; } }, TaskScheduler.FromCurrentSynchronizationContext());</pre>	点路径:	☞加压泵1给定 /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力 言添加的代码段如下 id button1_Click(object sender, System.Windows.RoutedEventArgs e) ODO: Add event handler implementation here. configuration = Application.Current.Resources["DataHub"] as IUaConnectio configuration == null !configuration.IsValid) { throw new ServiceResultException(StatusCodes.BadConfigurationError); connection = UaConnectionManager.Current.GetConnection(configuration session = (connection == null) ? null : connection.Connection; session == null) { throw new ServiceResultException(StatusCodes.BadNotConnected); variableId = new NodeId(102872U, 3);// /Objects/2:System/2:PlantObjects/	nCon)); 2:太原	节点ID: 确定 figuration; 取/2:站101A/2::	ns=3;i=1028 取消
<pre>} else { // TODO: OK // var dataValue = t.Result; } }, TaskScheduler.FromCurrentSynchronizationContext());</pre>	点路径: 〒确定) rivate vc { // 1 var if (var if (var if (var var ses	☞加压泵1给定 /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力 后添加的代码段如下 id button1_Click(object sender, System.Windows.RoutedEventArgs e) ODO: Add event handler implementation here. configuration = Application.Current.Resources["DataHub"] as IUaConnection configuration == null !configuration.IsValid) { throw new ServiceResultException(StatusCodes.BadConfigurationError); connection = UaConnectionManager.Current.GetConnection(configurationError); connection == null) ? null : connection.Connection; session == null) { throw new ServiceResultException(StatusCodes.BadNotConnected); variableId = new NodeId(102872U, 3);// /Objects/2:System/2:PlantObjects/ ion.ReadValueAsync(variableId).ContinueWith((t) => {	nCon)); /2:太原	节点ID: 确定 figuration; ī/2:站101A/2:2	ns=3;i=1028 取消 二次供水压力
// TODO: OK // var dataValue = t.Result; } }, TaskScheduler.FromCurrentSynchronizationContext());	点路径: 〒确定) rivate vc { // 1 var if (} var if (} var ses	☞加压至1给定 /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力 言添加的代码段如下 id button1_Click(object sender, System.Windows.RoutedEventArgs e) ODO: Add event handler implementation here. configuration = Application.Current.Resources["DataHub"] as IUaConnection configuration == null !configuration.IsValid) { throw new ServiceResultException(StatusCodes.BadConfigurationError); connection = UaConnectionManager.Current.GetConnection(configurationError); connection == null) ? null : connection.Connection; session == null) { throw new ServiceResultException(StatusCodes.BadNotConnected); variableId = new NodeId(102872U, 3);// /Objects/2:System/2:PlantObjects/ sion.ReadValueAsync(variableId).ContinueWith((t) => { if (t.Exception != null) { // TODO: Frror	nnCon)); /2:太原	节点ID: 确定 figuration; 氡/2:站101A/2:1	ns=3;i=1028 取消 二次供水压力
<pre>// var dataValue = t.Result; } }, TaskScheduler.FromCurrentSynchronizationContext());</pre>	点路径: 〒确定) rivate vc { // 1 var if (: } var if (: } var ses	PhileEq1台定 /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力 后添加的代码段如下 id button1_Click(object sender, System.Windows.RoutedEventArgs e) ODO: Add event handler implementation here. configuration = Application.Current.Resources["DataHub"] as IUaConnectio configuration == null !configuration.ISValid) { throw new ServiceResultException(StatusCodes.BadConfigurationError); connection = UaConnectionManager.Current.GetConnection(configuration session = (connection == null) ? null : connection.Connection; session == null) { throw new ServiceResultException(StatusCodes.BadNotConnected); variableId = new NodeId(102872U, 3);// /Objects/2:System/2:PlantObjects/sion.ReadValueAsync(variableId).ContinueWith((t) => { if (t.Exception != null) { // TODO: Error } else {	nnCon)); ?2:太原	节点ID: 确定 figuration; 取/2:站101A/2:1	ns=3;i=1028 取消 二次供水压力
} }, TaskScheduler.FromCurrentSynchronizationContext());	点路径: 亍确定丿 rivate vc { // 1 var if (} var if (} var ses	PhileEq1台定 /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力 后添加的代码段如下 id button1_Click(object sender, System.Windows.RoutedEventArgs e) ODO: Add event handler implementation here. configuration = Application.Current.Resources["DataHub"] as IUaConnectio configuration == null !configuration.IsValid) { throw new ServiceResultException(StatusCodes.BadConfigurationError); connection = UaConnectionManager.Current.GetConnection(configuration session = (connection == null) ? null : connection.Connection; session == null) { throw new ServiceResultException(StatusCodes.BadNotConnected); variableId = new NodeId(102872U, 3);// /Objects/2:System/2:PlantObjects/sion.ReadValueAsync(variableId).ContinueWith((t) => { if (t.Exception != null) { // TODO: Error } else { // TODO: OK	nnCon)); '2:太原	节点ID: 确定 figuration; ī/2:站101A/2:	ns=3;i=1028 取消 二次供水压力
} }, TaskScheduler. FromCurrentSynchronizationContext());	点路径: 〒确定丿 rivate vc { // 1 var if (} var if (} var ses	PhileEq1台定 /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力 言添加的代码段如下 id button1_Click(object sender, System.Windows.RoutedEventArgs e) ODO: Add event handler implementation here. configuration = Application.Current.Resources["DataHub"] as IUaConnection configuration == null !configuration.IsValid) { throw new ServiceResultException(StatusCodes.BadConfigurationError); connection = UaConnectionManager.Current.GetConnection(configurationError); connection == null) ? null : connection.Connection; session == null) { throw new ServiceResultException(StatusCodes.BadNotConnected); variableId = new NodeId(102872U, 3);// /Objects/2:System/2:PlantObjects/ sion.ReadValueAsync(variableId).ContinueWith((t) => { if (t.Exception != null) { // TODO: Error } else { // TODO: OK // var dataValue = t.Result;	nCon)); /2:太原	市点ID: 确定 figuration;	ns=3;i=1028 取消 二次供水压力
j, raskouleuren omeanen synen omzation context(),	点路径: 〒确定丿 rivate vcc { // 1 var if (} var if (} var ses	PhileEq1台定 /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力 言添加的代码段如下 id button1_Click(object sender, System.Windows.RoutedEventArgs e) ODO: Add event handler implementation here. configuration = Application.Current.Resources["DataHub"] as IUaConnection configuration == null !configuration.IsValid) { throw new ServiceResultException(StatusCodes.BadConfigurationError); connection = UaConnectionManager.Current.GetConnection(configurationError); connection = UaConnectionManager.Current.GetConnection(configurationError); connection == null) ? null : connection.Connection; session == null) { throw new ServiceResultException(StatusCodes.BadNotConnected); variableId = new NodeId(102872U, 3);// /Objects/2:System/2:PlantObjects/ sion.ReadValueAsync(variableId).ContinueWith((t) => { if (t.Exception != null) { // TODO: Error } else { // TODO: OK // var dataValue = t.Result;)); /2:太原	市点ID: 确定 figuration; 夏/2:站101A/2:2	ns=3;i=1028 取消 二次供水压力
	点路径: 亍确定) rivate vc { // 1 var if (var if (var if (var if (<pre></pre>	nCon)); 2:太原	市点ID: 确定 figuration; ₹/2:站101A/2:1	ns=3;i=1028 取消 二次供水压力
	点路径: 示确定) rivate vcc { //1 var if (var if (var if (var if (var if () var if () var jess }, T.	<pre></pre>	nCon)); 2:太原	市点ID: 确定 figuration; 夏/2:站101A/2:1	ns=3;i=1026 取消 二次供水压力

```
private void button1_Click(object sender, System.Windows.RoutedEventArgs e)
  {
      // TODO: Add event handler implementation here.
      var configuration = Application.Current.Resources["DataHub"] as IUaConnectionConfiguration;
      if (configuration == null || !configuration.IsValid) {
          throw new ServiceResultException(StatusCodes.BadConfigurationError);
      }
      var connection = UaConnectionManager.Current.GetConnection(configuration);
      var session = (connection == null) ? null : connection.Connection;
      if (session == null) {
          throw new ServiceResultException(StatusCodes.BadNotConnected);
      }
      var variableId = new NodeId(102872U, 3);// /Objects/2:System/2:PlantObjects/2:太原/2:站101A/2:二次供水压力
      session.ReadValueAsync(variableId).ContinueWith((t) => {
          if (t.Exception != null) {
              // TODO: Error
          } else {
              // TODO: OK
              this.textBlock1.Text = t.Result.ToString();
          }
      }, TaskScheduler.FromCurrentSynchronizationContext());
```

```
}
```

点击运行测试。

MainWindow		
修改变量值	4.523	
	4.523	
读取变量值	4.523	

5.2.4 订阅变量实时值(Subscribe variable's realtime data)

订阅变量实时值完成之后,当此变量发生变化时,会触发值变化通知事件,由事件参数 获取变量的新值。

在按钮 Click 方法中,右键选择插入代码段->订阅变量实时值。

⋇	剪切	Ctrl+X		
	复制	Ctrl+C		
2	粘贴	Ctrl+V		
\boldsymbol{x}	刪除	Del	*	设置相对绑定对象
	添加/删除注释 缩进		*	读取变量实时值 修改变量当前值
	重命名	F2	*	订阅变量实时值 读取变量历史数据
•	跳转到定义	F12	*	解析节点路径为ID
	插入代码片段	•	*	调用方法

选择需要订阅的变量的节点;

选择节点		– 🗆 ×
连接名称: DataHub	刷新	新建
🔺 🦢 Root		<u>^</u>
Objects		
🕨 😚 Server		
🔺 🧁 System		
Alarms		
A Global Methods		
PlantObjects		
▲ 🔧 站101A		
▲ 补水泵1频率反馈		
☆ 小水泵1运行状态 ☆		
▲ 小水泵2运行状态		
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□		
◎ ***		
☆ 加压泵1反馈		
→ 加压范1台定		-
节点路径: /Objects/2:System/2:PlantObjects/2: 太原 /2:站101A/2:二次供水压力	节点ID:	ns=3;i=102872
	确定	取消

点击 OK 后,如图,选中的蓝色背景部分为 Notification 事件触发的方法。DataValue 类型的对象 v 为此变量的数据类示例。



由于事件参数 e 会有重名冲突,我们将此事件进行修改。



单独定义一个通知事件触发执行的方法 m_Notification().并将此变量的值赋值给新定义的 string 类型的 monitoredItemValueString.然后在新添加的按钮中定义显示此变量值的 Click 方法,将值在新添加的 TextBlock 中显示出来。



AicStudio 用户手册

MainWindow		
	22	_
修成受重值	33	
	33	
	22	
读取变量值	55	
订阅变量值	33	显示值
		t

5.2.5 读取变量历史数据(Read variable's history data)

读取历史数据的时候,需要 DataHub 服务器上<mark>配备密码锁</mark>,给予历史数据查看和取出 权限。

以下为读取变量历史数据的示例。新建 WPF 工程。

新建工程		– 🗆 ×
分类		模板
🔺 ն C#		BawPF Application
— 🚞 Si	ilverlight	聽WPF Control Library
— 🚞 W	Veb	
N 🛄 🛛	VPF	
A project for	r creating rich desktop	o applications that run on Windows.
名称 F	ReadHistoryData	
路径(C:\Users\dst\Document	ts\AicStudio\Projects\WpfApplication2\
框架版本 4	1.5	▼
		ゆ 中 一 四 当
		194 <u>2</u> <u>4</u> 2/H

在 Xaml (标记文本) 界面添加如下控件, 2 个 DatePicker 用来选择读取历史数据的起始 时间和结束时间,一个 TextBox 用来输入读取的数据量(自起始时间开始选中的 DataHub 节 点存储的历史数据的个数),一个 Button 用于在单击时触发读取历史数据,一个 ListBox 用于显示读取到的所有数据。

MainWindow			
起始时间	选择日期	15	
结束时间	选择日期	15	
数据量(1-100)			
读取历史数据			

添加按钮单击事件方法,在按钮(读取历史数据)的 Click 方法中,右键单击选择插入 代码片段->读取变量历史数据。

¥	剪切	Ctrl+X		
Ð	复制	Ctrl+C		
	粘贴	Ctrl+V		
X	刪除	Del	*	设置相对绑定对象
_	添加/删除注您		*	读取变量实时值
_			*	修改变量当前值
	SIH JT			订阅变量实时值
	重命名	F2		读取变量历史数据
•	跳转到定义	F12	*	解析节点路径为ID
	插入代码片段	+	*	调用方法

然后选择要读取的变量节点

选择节点		– 🗆 ×
连接名称: DataHub 🔹	刷新	新建
A 📴 Root		
Objects		
🕨 😤 Server		_
🔺 🦙 System		
Alarms		
A GlobalMethods		
PlantObjects		
▲ 🔧 太原		
▲ ⁴ \$ 站101A		
🚰 补水泵1频率反馈		
🚰 补水泵1运行状态		
🚰 补水泵2频率反馈		
🚰 补水泵2运行状态		
🚰 补水泵3频率反馈		
🚰 补水泵3运行状态		
▶ 11 二次供水温度		
▶ 🍕 二次供水温度报警		
▶ 叠二次供水压力		
▶ 😚 二次供水压力报警		
▶ 😁 二次回水温度		
▶ mm 二次回水压力		
1 功率		
1 加压泵1本地远程		
11 加压泵1反馈		
		*
卫品路径: /Ubjects/2:System/2:PlantUbjects/2: 太原 /2:延101A/2:二次供水压力	节点ID:	ns=3;i=102872
	确定	取消

点击确定后,即添加代码段完成,然后进行对代码段的修改。

修改一、在建立连接时,线程挂起1秒(如果调试出现异常,可将此时间延长),等待建立连接的线程连接成功。(若工程已有相关 OPC UA 的相对绑定或者绝对绑定,那么连接会被默认建立,此时不需要将线程进行挂起)

```
var connection = UaConnectionManager.Current.GetConnection(configuration);
Thread.Sleep(1000);
var session = (connection == null) ? null : connection.Connection;
if (session == null)
{
    throw new ServiceResultException(StatusCodes.BadNotConnected);
}
```

修改二、将起始时间和结束时间对应到相应的控件上去, startDate 和 endDate 分别是 2 个 DatePicker 控件的名称。

var startTime = (DateTime)**this**.startDate.SelectedDate; var endTime = (DateTime)**this**.endDate.SelectedDate;

//var startTime = DateTime.UtcNow.Subtract(TimeSpan.FromMinutes(60)); // **TODO**: replace with real start time //var endTime = DateTime.UtcNow; // **TODO**: replace with real end time

修改三、将 TextBox 的输入值作为数据量参数,最大值为 100

```
session.HistoryReadRawDataAsync(

new NodeId[] { variableId },

startTime,

endTime,

UInt32.Parse(this.dataNum.Text), // maximum records per variable of each call

false,

null).ContinueWith((t) => {
```

修改四、在处理数据(Process Data)处,将读取的数据以字符串集合(List<string>)的形式作为 ListBox 控件的 ItemSource,使数据在 ListBox 上显示出来。

```
dataValues = result.Result:
    if (dataValues == null || dataValues.Count == 0) {
        // TODO: NO DATA
    } else {
        // TODO: Process Data
List<string> listString = new List<string>();
if (dataValues != null)
{
  foreach (DataValue dv in dataValues)
  {
     listString.Add(dv.Value.ToString());
  }
}
this.listValue.ItemsSource = listString;
    }
}
```

修改完毕后点击运行,进行测试。测试需要 DataHub 已经存储了历史数据,并打开 DataHub 服务。

MainWindow	-		-	
起始时间	2014/12/18	15		
结束时间	2014/12/20	15		
数据量(1-100)	10			
读取历史数据	50.87262 51.74498 52.6168 53.48782 54.35779 55.22642 56.09347 56.95866			
5.2.6 解析节点路径为 ID(Translate path)

新建工程				– 🗆 ×					
分类		模板							
🔺 ն C#		UWPF Application							
	Silverlight	INVERTIGATION LIBRARY							
	Web								
	WPF								
A project	or creating rich deskto	applications that run on Windows.							
夕称	TranslatePath								
·口小·									
路佺	C:\Users\dst\Documen	ts\AicStudio\Projects\WpfApplication2\							
框架版本	4.5			*					
			确定	取消					
在 Xaml(标	记文本)界面添加控件	- o							
Mai	nWindow.								
	Ĩ								
	- 82								
		<u></u> 1							
9 — /2									
	NodeID								

先建 WPF 工程进行此代码段的测试。

在按钮的 Click 方法中插入代码段->解析节点路径为 ID

¥	剪切	Ctrl+X		
	复制	Ctrl+C		
	粘贴	Ctrl+V		
X	刪除	Del	æ	设置相对绑定对象
_	沃加/删除注露		æ	读取变量实时值
_	/까//॥/ ┉/짜/土木井		*	修改变量当前值
	31H.J.T.			订阅变量实时值
	重命名	F2		读取变量历史数据
-	跳转到定义	F12		解析节点路径为ID
	插入代码片段	•	æ	调用方法



连接名称: DataHub 🔹 刷線	f 新建
🔺 🗁 Root	*
Dijects	
▶ 😤 Server	
🔺 🗁 System	
🕨 🛅 Alarms	
GlobalMethods	
PlantObjects	
▲ 🔧 站101A	
計算者水源1频率反馈	
🚰 补水泵1运行状态	
III 补水泵2频率反馈	
☆ 补水泵2运行状态	
🚰 补水泵3频率反馈	
☆ 补水泵3运行状态	
▶ 1 二次供水温度	
▶ 🎌 二次供水温度报警	
▶ 🈚 二次供水压力报警	
▶ 1 二次回水温度	
▶ 計二次回水压力	
加压泵1本地远程	
12 加压泵1反馈	
	*
卫品路径: /Ubjects/2:System/2:PlantUbjects/2: 太 県/2:56101A/2:二次供水比刀	っ尻U: ns=3;i=102872
确定	取消

点击确定添加代码段

AicStudio 用户手册



MainWindow	
解析节点路径为ID	
NodeID	ns=3;i=102872

5.2.7 调用方法(Call method)

这里调用的方法是在 DataHub 中数据类型中定义的方法,用于修改或者读取数据类型 对象的变量值(更多详细的介绍请参见 AicDataHub 用户手册)。

下面是调用方法的示例。	首先新建 WPF 工程	CallMethodDemo;
-------------	-------------	-----------------

新建工程			- 🗆 ×
分类	模板		
🔺 隘 C#	巴 WPF Application		
— 🦳 Silverlight	WPF Control Library		
- C Web			
WPF			
<u></u>			
A project for creating rid	n desktop applications that run on Windows.		
名称 CallMethodE	emo		
路径 C:\Users\dst	Documents\AicStudio\Projects\WpfApplication2\		
框架版本 4.5			*
		梅宁 田	34
		UPPALE AX7	Ħ

在 XAML 用户界面(标记文本)上添加示例所用的控件,2个按钮分别用于关门和开门,一个 TextBlock 用于显示门的状态。



关门按钮 Click 方法中,选择 DataHub 服务器(此实例用的是电梯的类型),选择对象 Elevator 的方法节点 CloseDoor,点击 OK,添加关门的方法代码段。

选择节点				– 🗆 ×
连接名称:	Server2	*	刷新	新建
	► ?	\$ Crane3		*
	4	\$ Elevator1		
		🚰 BeginTime		
	►	=🖗 ChangeState		
	►	≠♥CloseDoor		
		CloseDoorClick		
		2 CurFloor		
		m Direction		
		m DoorClose		
		PoorClosePause		
		🚰 DoorOpen		
		🚰 DoorOpenStart		
		mageShow ???		
		🚰 InnerLight		
	►	=•>InnerPush		
		🚰 LastTime		
	►	=•• OpenDoor		
		Provider Floor Array		
		🚰 OrderTypeArray		
		🚰 OutDownLight		
	►	=•• OutDownPush		
		🚰 OutUpLight		
	►	=•• OutUpPush		
		🚰 Speed		
		Timing		*
节点路径:	/Object	s/2:System/2:PlantObjects/2:Elevator1/2:CloseDoor	节点ID:	ns=3;i=31294
			确定	取消

此代码由于没有输出参数,所以不需要进行进一步的修改。



开门按钮的 Click 方法添加和关门类似,选择节点时选择 Elevator1 对象下的 OpenDoor 方法 节点。

选择节点			– 🗆 ×
连接名称:	Server2 v	刷新	新建
-	riantobjects		*
	AreaEnvironment1		
	▶ 4\$ BZ1		
	▶ < <p>4\$ BZ2</p>		
	▶ 🐴 BZ3		
	▶ 🐴 Crane1		
	🕨 🔩 Crane2		
	🕨 🔩 Crane3		
	🔺 🔧 Elevator1		
	🚰 BeginTime		
	▶ =∳ ChangeState		
	▶ =�CloseDoor		
	🚰 CloseDoorClick		
	🚰 CurFloor		
	main Direction		
	🚰 DoorClose		
	ToorClosePause		
	🚰 DoorOpen		
	🚰 DoorOpenStart		
	ImageShow		
	🚰 InnerLight		
	▶ =∲InnerPush		
	🚰 LastTime		
	► <\$OpenDoor		
	TorderFloorArray		-
节点路径:	/Objects/2:System/2:PlantObjects/2:Elevator1/2:OpenDoor	节点ID:	ns=3;i=31299
		确定	取消

然后将 TextBlock.Text 属性 OPC UA 绝对绑定到 Elevator1 对象的 DoorOpen 变量上,用于显示当前的电梯门的状态。(绑定的细节参见 4.2 节数据绑定)

为 textBlock.Text 创建数据绑定 绑定类型 OPC UA绝对绑定	- 0
连接名称: Server2 新建	属性 口显示匹配类到
注接名称: Server2 ● 刷新 新建 ● ③ BZ2 ● ③ BZ3 ● ④ Crane1 ● ④ Crane2 ● ④ Crane3 ● ④ CloseDoor □ ClosePause □ DoorClose □ DoorClose □ DoorClose □ DoorClose □ DoorClose □ ClosePause □ Clos	 ■ 位显示匹配换到 ✓ UaVariable AccessLevel: (Byte) ArrayDimensions: (UInt32) BrowseName: (QualifiedName) DataType: (NodeId) DataType: (NodeId) DataType: (LocalizedText) DisplayName: (LocalizedText) DisplayName: (LocalizedText) DisplayName: (Boolean) IsGoal (Boolean) IsGoal : (Boolean) LocalServerTimestamp: (DateTime) MinimumSamplingInterval : (Double) NodeId: (NodeId) NodeId: (VaNedPath) ServerTimestamp: (DateTime)
T LastTime ► OpenDoor OrderFloorArray OrderTypeArray OutDownLight ► OutDownPush OutDupLight T 点路径: /Objects/2:System/2:PlantObjects/2:Elevator1/2:DoorOpen	 ▶ SourceTimestamp : (DateTime) ▶ StatusCode : (StatusCode) Tag : (Object) UserAccessLevel : (Byte) UserWriteMask : (UInt32) Value ank : (Int32) ValueRank : (Int32) 郑定路径 Value
▼ 更多设置	确定 取消



MainWindow	
关门	
开门	
门的状态 true	

6 编译与调试

在 AicStudio 的主页菜单中有下图所示功能区: 支持 Debug 和 Release, Debug 是调试用的,可以打断点执行, Release 是发布用的,不能打断点执行。下图中 第一行的四个小按钮从左到右依次是: 生成解决方案; 生成选中的工程(在解决 方案浏览器中选中想要生成的工程); 调试模式运行(断点有效); 不调试运行(断 点无效)。

🚟 🛗 🕨	₽
Debug	•
Any CPU	•
生成	

工程菜单中有如下常用的编译和调试功能,基本和 VisualStudio 中的常用调试功能类似。

_	主页	工程	视图	工具											
2000年 生成解決	大家	之 重新生成解	决方案	清理解决方案	生成	全 重新生成	清理	调试	▶ 运行 (不调试)	停止调试	中断	建续	永 逐过程	逐语句	部出
		生成解决	方案			生成工程				调调	式/运行				

7 快速入门示例

本节中,将使用 AicStudio 新建一个 WPF 项目,结合 DataHub 和 IOServer 具体实现一个简单的监控工。在此简略说明一点: IOServer 是用于现场采集数据的,并将数据主动或者被动的发送给 DataHub, DataHub 进行对数据的存储、数据下发(下发给 IOServer)作为 UI 界面的数据源(通过界面 Binding 至 DataHub 某对象实例的某变量), IOServer 和 DataHub 的具体可以查看 AicTech 的 IOServer 用户手册和 DataHub 用户手册。

7.1 使用 DataHub 建立对象模型和对象实例

打开 AicTechDataHub Configurator,打开之前应确保 AicTechDataHub Server 正在运行,可以在电脑的服务中查看开启。下图登陆用户名、密码都为 sa。

登录		_ 🗆 X
服务器地址	opc.tcp://127.0.0.1:4520	
安全模式	None	•
安全策略	None	•
用户名	sa	
密码	••	
	确定	取消

新建对象类型,右击模型浏览器中的对象类型选择新建对象类型如下图。

模型浏览器			φ×
ロエノオ	象		
自动会光	πII		
		复制	Ctrl+C
	Ж	剪切	Ctrl+X
▶ 🚞安全	<u></u>	粘贴	Ctrl+V
	2	刷新	
	2	新建对象	类型

自定义浏览名称,如下图。

新建对象类型		-		х
浏览名称	TransportCar			
描述				
	确定	取消	í	

对于运输车这个模型,这里简略定义这样几个变量:车速 Speed,运输车钢 罐中物料的物位,钢罐内部的温度(也应该有压力,这里省略)。

右击新建完成的 TransportCar 选择新建变量,如下图。建 Speed 变量, MaterialLeval 变量, Tempurature 变量,数据类型都为 Double 即可。

4	🔁 对象类型			复制	Ctrl+C
	🗠 Transpo	ortCar	Ж	剪切	Ctrl+X
•	┛报警		B	粘贴	Ctrl+V
•	一脚本		-		
•	🔤 安全		-		
			ŝ		
			25	解除到父 刘 家的大助	c .
				編辑	
			^₿	新建对象	
				新建目录	
			2	新建变量	
			Ŷ	新建属性	
ł	新建变量				_ 🗆 X
	浏览名称	Speed			
	描述	车速			
	数据类型	Double			
	维度	Scalar			•
	保存历史数据				
	类型定义	DataItemT	ype		
	建模规则	Mandatory			•
	启用批量模式				
				确定	取消
新建完成如下图:					
		对象类型 □ ☆ Trans - ☆ M - ☆ S ☆ T	sport later peed empe	Car ialLevel erature	

右击工厂对象(PlantObject)选择新建对象(如下图),类型定义选择刚才新建的 TransportCar,新建两个名为 Car1 和 Car2 的对象实例。

模型浏览器		φ×	Transpo
「二工厂対象		复制	Ctrl+C
▲ 🧰 对象类型	X	前切	Ctrl+X
🖌 🛂 TransportCar		35 %3 米5 05	Ctrl+V
— 😁 MaterialL	.e	11121	
- 🚰 Speed	2	刷新	ā,
- 🚰 Temperat	tu 🔧	新建对象	ż
▶ 📴 报警		新建目录	ſ
▶ 🚞 脚本			
新建对象			_ □
浏览名称 Car1	_		
描述	单	击此处	选择类型
事件通知标志 1			

新建对象			х
浏览名称	Car1		
描述		单击此处选择类型	
事件通知标志	1		
类型定义	TransportCar		
建模规则	None		•
启用批量模式			
		确定取消	

新建完成后如下图:

模型浏览器	þ	×
▲ 🗀 工厂对象		
🖌 🔩 Car1		
MaterialLevel		
- 😁 Speed		
🗌 🚰 Temperature		
🕨 🕂 Car2		

至此运输车的对象模型和实例均已建立完成,后续的工作就是将 IOServer 中 采集上来的对应数据和 DataHub 实例中对应的变量相互绑定以及将 UI 界面上的 变量和 DataHub 对象模型中对应的变量进行绑定。

7.2 使用 IOServer 为 DataHub 提供数据

打开 IOServer (以管理员方式),新建工程如下图:

新建工程			×	Ľ
	工程名称	TransportCarDataSource]	
	路径	C:\ProgramData\AicTech\IOServer\Projects		
		确定	取消	

点击确定,右击新建的工程名选择修改工程,右边的窗口出现此工程信息(如下 图),在主动上发出打上勾后目标服务器和缓存转发设置被激活。

G	TransportCarDataSource 工程信息 目标服务器 缓存转发设置 工程名称 TransportCarDataSource 主动上发 在此处打上勾				
	工程信息	目标服务器	缓存转发设置		
		工程名称 🔳	TransportCarDataSource		
		主动上发			
		在此	处打上勾		
1					

点击目标服务器,配置如下。

E程信息 目标服务器 鎖 OPC UA服务器信息	爰存转发设置 			
服务器地址	opc.tcp://hph:4520/Datal	lubServer		
安全模式	None			测试连接
安全策略	None	点击配置服务地址	-	•
数据发布间隔			1,000 🛔	室秒
会话超时		60	0,000 🚔	室秒
保持活动连接的间隔			5,000 🚔	室秒
保持活动连接的错误阈值			3 🛔	•
使用二进制编码	\checkmark			
验证设置				
) 唐名				_
▶ 用户名	sa	密码●●		
〕 证书	<错误的证书>			
) 令牌				

点击配置服务器地址按钮,窗口如下,点击右边红框的按钮,选择红框条目,其 中 hph 是本人的计算机名,不同计算机名字可能不同,点击确定。

发现服务器	- - X
主机名 Hph	•
服务器列表 AicTech DataHub Server	终结点列表 opc.tcp://hph:4520/DataHubServer - SignAndEncrypt opc.tcp://hph:4520/DataHubServer - Sign opc.tcp://hph:4520/DataHubServer - None http://hph:4521/DataHubServer - SignAndEncrypt http://hph:4521/DataHubServer/Basic256 - Sign http://hph:4521/DataHubServer/None - None 确定 取消

目标服务器配置窗口中的验证设置选择用户名 sa, 密码 sa。

右击工程名选择新建通道,配置如下图,通道名称可自定义,通道类型选择 Simulation 驱动类型选择 Simulation,点击确定。在此说明一点:Simulation 驱动 是一个模拟的数据来源,可以按照一定过得规律产生模拟数据,实际中需要从现 场的设备中采集数据,还需要开发相应的 IOServer 驱动,具体见 IOServer 驱动开 发文档。

新建通道		-		х
通道信息				
通道名称	TransportCarChannel			
描述				
通道类型	Simulation		•	
驱动类型	Simulation		•	
恢复间隔		30	*	秒
	确定	取消	ij	

右击通道名称选择新建设备,如下图,设备名称可自定义,设备数据中最大值和 最小值在此选择默认的即可,变化周期用于设置数值大小变化1所间隔的时间, 按默认设置即可(1000ms)也可自行设置。

nsportCar1			
	3	‡ 道	e续失败
	60	\$ 원	þ
	86,400	÷ 10	þ
	0.0	0	
	100.0	00	
	1,00	00	室秒

右击设备名称选择新建变量组,变量组名称自定义。

新建变量组	х
组名	TransportCarGroup1
描述	
	确定取消

为方便起见,我们选择则从 DataHub 导入变量(当然也可以自行添加相应的变量), 右击变量组名选择从服务器导入,在弹出的窗口进行如下选择,点击确定即可。

选择项		х
- 节点树		
ØDjects		
⊢ 🏤 Server		
🖛 🦢 System		
► 🔁 Alarms		
▶– 🔧 GlobalMethods		
🖌 🗁 PlantObjects		
🖌 🕂 Car1		
— 🚰 MaterialLevel		
— 🚰 Speed		
- 🚰 Temperature		
▶ 🕂 🔶 🔶 🔶		
► 🔁 Scripts		
► 🔁 Security		
确定	消	

导入后结果如下图:



导入成功后再设置一下各个变量模拟采集的方式。双击变量名,右边会显示 变量模拟采集的设置窗口,变量数据窗口中的寄存器名称选项用于设置变量模拟 采集方式,此处将其选择设置为三角波形式"Triangle",这样变量的值会从最小 值以 1/s 的变化频率增加到最大值之后再从最大值均匀的减小到最小值,循环 进行,此处我对这三个变量的寄存器名都设置为"Triangle",可自定义。

至此我们已经配置完 TransportCar1 的变量了,类似我们在 TransportCarChannel下再建一个设备 TransportCar2 将在 DataHub 中的 Car2 这个 对象实例中的变量导入到 IOServer 中,为了以示区分,可以改变变量模拟采集方 式,这里对 TransportCar2 的变量的模拟采集方式改为"Increase"、修改 TransportCar2 将最小值改为 20,最大值改为 90,变化周期改为 500ms,可自定 义。



导入完成后点击保存(必须先保存),右击工程名选择"设为启动工程(本机)"(若设置失败,那么可能是打开 IOServer 时不是以管理员方式启动的)。在服务中开启 IOServer,若 IOServer 服务正在运行则需重启服务(此时也应该已经打开 DataHub 服务)。完成后 DataHub 中 Car1 和 Car2 两个实例中的变量已经随着 IOServer 中相应的变量的变化而变化了。从 4.2 中我们已经知道 UI 界面上控件的属性已经可以绑定到 OPC UA Data Node (DataHub 中已经定义好的变量),这样就可以在 UI 界面上实时显示我们需要监测的变量。

7.3 使用 AicStudio 构建监控画面

打开 AicStudio 新建 WPF 项目,命名为 "AicTech 快速入门"从工具窗口拖入艾克信控控件: HorizontalLinearGauge, Fans003, VesselTank, RacingGauge,对每个控件可以用上文中的特效修饰。HorizontalLinearGauge 用于表示温度,RacingGauge 用于表示车速,VesselTank 用于 表示钢罐,控件本身可以显示钢罐的物位,Fans003 显示的风扇用于表示制冷机,其功能为 在温度高于 40 时进行制冷(叶扇的转动),并且温度越高,叶扇旋转越快。



在工具窗口处选择 OPC UA 服务浏览器添加一个新的连接:



此处添加的而服务器连接和上文 IOServer 中配置目标服务器相同:

lost Name Hph		•
erver List	Endpoint List	
icTech DataHub Server	opc.tcp://hph:4520/DataHubServer - SignAndEncrypt	
icTech IOServer	opc.tcp://hph:4520/DataHubServer - Sign	
	opc.tcp://hph:4520/DataHubServer - None	
	http://hph:4521/DataHubServer - SignAndEncrypt	
I	http://hph:4521/DataHubServer/Basic256 - Sign	
	http://hph:4521/DataHubServer/None - None	
	OK	Cancel
	OK	Cancel
Opc Ua Connection Config	OK uration ^R ookmark Build Language/The	Cancel
Opc Ua Connection Configu	OK uration ^{Pookmark} Build Language/Them	Cancel
Opc Ua Connection Configu OPC UA Server Information Connection Name	OK uration ^{Bookmark} Build Language/Them n AicTech DataHub Server	Cancel
Opc Ua Connection Configu OPC UA Server Information Connection Name Description	OK uration ^{500kmark} Build Language/Thee n AicTech DataHub Server	Cancel
Opc Ua Connection Configu OPC UA Server Information Connection Name Description Server Uri	OK uration ^{Pookmark} Build Language/Thee n AicTech DataHub Server opc.tcp://hph:4520/DataHubServer	Cancel
Opc Ua Connection Configu OPC UA Server Information Connection Name Description Server Uri Security Mode	OK uration ^{Dookmark} Build Language/Thee n AicTech DataHub Server opc.tcp://hph:4520/DataHubServer None	Cancel
Opc Ua Connection Configu OPC UA Server Information Connection Name Description Server Uri Security Mode Security Policy	OK uration ^{Dookmark} Build Language/Thee n AicTech DataHub Server opc.tcp://hph:4520/DataHubServer None	Cancel
Opc Ua Connection Configu OPC UA Server Information Connection Name Description Server Uri Security Mode Security Policy Publishing Interval	OK uration ^{® ookmark} Build Language/Thee n AicTech DataHub Server opc.tcp://hph:4520/DataHubServer None 1,000	Cancel
Opc Ua Connection Configu OPC UA Server Information Connection Name Description Server Uri Security Mode Security Policy Publishing Interval Session Timeout	OK uration ⁵ ookmark Build Language/Thee n AicTech DataHub Server opc.tcp://hph:4520/DataHubServer None 1,000 600,000	Cancel
Opc Ua Connection Configu OPC UA Server Information Connection Name Description Server Uri Security Mode Security Policy Publishing Interval Session Timeout KeepAlive Interval	OK uration [®] ookmark Build Language/Thee n AicTech DataHub Server opc.tcp://hph:4520/DataHubServer None 1,000 600,000 5,000	Cancel
Opc Ua Connection Configu OPC UA Server Information Connection Name Description Server Uri Security Mode Security Policy Publishing Interval Session Timeout KeepAlive Error Threshold	OK uration ookmark Build Language/Thee n AicTech DataHub Server opc.tcp://hph:4520/DataHubServer None 1,000 600,000 5,000 3	Cancel
Opc Ua Connection Configu OPC UA Server Information Connection Name Description Server Uri Security Mode Security Policy Publishing Interval Session Timeout KeepAlive Interval Like Binary Encoding	OK uration Cokmark Build Language/Thee AicTech DataHub Server opc.tcp://hph:4520/DataHubServer None 1,000 600,000 5,000 3	Cancel
Opc Ua Connection Configu OPC UA Server Information Connection Name Description Server Uri Security Mode Security Policy Publishing Interval Session Timeout KeepAlive Interval KeepAlive Error Threshold Use Binary Encoding	OK uration Bookmark Build Language/Thee AicTech DataHub Server opc.tcp://hph:4520/DataHubServer None 1,000 6600,000 5,000 3	Cancel
Opc Ua Connection Configu OPC UA Server Information Connection Name Description Server Uri Security Mode Security Policy Publishing Interval Session Timeout KeepAlive Interval KeepAlive Interval Use Binary Encoding	OK urationPookmark Build Language/Thee n AicTech DataHub Server opc.tcp://hph:4520/DataHubServer None 1,000 600,000 5,000 3	Cancel
Opc Ua Connection Configu OPC UA Server Information Connection Name Description Server Uri Security Mode Security Policy Publishing Interval Session Timeout KeepAlive Interval KeepAlive Interval KeepAlive Error Threshold Use Binary Encoding Authentication Settings	OK urationPookmark Build Language/Thee n AicTech DataHub Server opc.tcp://hph:4520/DataHubServer None 1,000 600,000 5,000 3	Cancel
Opc Ua Connection Configu OPC UA Server Information Connection Name Description Server Uri Security Mode Security Policy Publishing Interval Session Timeout KeepAlive Interval KeepAlive Error Threshold Use Binary Encoding Authentication Settings O Anonymous O User Name	OK uration okmark Build Language/Thee n AicTech DataHub Server opc.tcp://hph:4520/DataHubServer None 1,000 6600,000 5,000 3 3 3 3 3	Cancel
Opc Ua Connection Configu OPC UA Server Information Connection Name Description Server Uri Security Mode Security Mode Security Policy Publishing Interval Session Timeout KeepAlive Interval KeepAlive Interval KeepAlive Error Threshold Use Binary Encoding Authentication Settings Authentication Settings User Name Certificate	OK uration ookmark Build Language/Thee n AicTech DataHub Server opc.tcp://hph:4520/DataHubServer None 1,000 600,000 5,000 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3	Cancel

点击确定后双击这个新建的连接,这时在 AicStudio 中已经可以看到之前新建的 两个实例:



在浏览器窗口右击项目名称,选择"Add New Item",添加一个类,命名为 T2ColorConverter,如图。

New ItemColorConverter	Y OConvert 🕒 🗖 🔀
Categories	Templates
🔺 <u> </u> C#	Application Configuration File
— 🦲 General	EApplication Manifest File
— 🧰 Silverlight	Assembly Information File
WPF	4 Class
	⊶OInterface
	🖪 Resource File
	Text File
	🔊 Xml File
Create an empty class declaratio	n.
Name T2ColorConverter	
	OK Cancel

在 T2ColorConverter 文件中添加引用 usingSystem.Windows.Data;

😫 T2	ColorConverter
1	using System;
2	using System.Collections.Generic;
3	using System.Linq;
4	using System.Text;
5	using System.Threading.Tasks;
6	using System.Windows.Data;
7	

在 T2ColorConverter 文件中添加如下代码:



此文件实现了一个值转换器,用于 HorizontalLinearGauge 控件在温度低于 40 度时温度 条的颜色显示绿色,40 到 80 之间显示橘黄色,高于 80 显示红色。

接下来将 DataHub 中的值绑定到特定控件的属性中:

对 VesselTank 控件的 Value 属性进行数据绑定:选择 OPCUA 绝对绑定,点击刷新 按钮,绑定到已经建好的实例 Car1 中的 MaterialLevel 属性的 Value 属性:

Create Data Binding for [VesselTank].Value				_ 0 <u>X</u>
Binding Type: OPC UA Data Node				•
Connection Name: AicTech_DataHub_Se Refresh New efresh New Refresh New Refresh New Refresh New Refresh New Refresh New Refresh New Refresh New Refresh New Refresh New Refr	Prope	ties: ISV000 : (BOOIEAN) ISValid : (Boolean) LocalServerTimestamp : (DateTime) MinimumSamplingInterval : (Double NodeClass : (NodeClass) Nodeld : (NodeClass) Nodeld : (NodeClass) NodePath : (UaNodePath) ServerTimestamp : (DateTime) SourceTimestamp : (DateTime) StatusCode : (StatusCode) Tag : (Object) UserAccessLevel : (Byte) UserWriteMask : (UIn32) Value : (Object) Value Akk : (UIn32)	Only display (natching types
 More settings 				
			ОК	Cancel

同样,将 RacingGauge 控件的 Value 属性绑定到 Car1 中的 Speed 属性的 Value 属性:将 HorizontalLinearGauge 控件的 Value 属性绑定到 Car1 中的 Temperature 属性的 Value 属性。

现在实现上文值转换器的功能:我们为 HorizontalLinearGauge 控件的 FluidBrush 属性创建一个数据绑定:绑定类型选择 RelativeSource Self,去掉仅显示匹配类型的勾(否则不能选择 Value 属性),选择 Value 属性,点开更多设置选择 T2ColorConverter 这个转换器(如果没有这个选项的话选择增加值转换器来创建 一个 T2ColorConverter 即可、检查之前有没有正确写完这个转换器,)。

Create Data Binding for [HorizontalLinearGauge].Flu	dBrush
Binding Type: RelativeSource Self	
	Properties: Dnly display matching typ
Bind [HorizontalLinearGauge] to itself.	TickPlacement : (TickPlacement)
	Ticks : (Double)
	ToolTip : (Object)
	TouchesCaptured : (IEnumerable <touchdevice>)</touchdevice>
	TouchesCapturedWithin : (IEnumerable < TouchDevice>)
	TouchesDirectlyOver : (IEnumerable <touchdevice>)</touchdevice>
	TouchesOver : (IEnumerable <touchdevice>)</touchdevice>
	Triggers : (TriggerBase)
	Uid : (String)
	UseLayoutRounding : (Boolean)
	Value : (Double)
	VerticalAlignment : (VerticalAlignment)
	VerticalContentAlignment : (VerticalAlignment)
	Path: Value
 Fewer settings 	
Converter T2ColorConverter	BindsDirectlyToSource
Converter Parameter:	IsAsync
Converter Parameter.	□ NotifyOnSourceUpdated
Binding Mode: Default	NotifyOnTargetUpdated



现在我们再来实现一个效果,让 Fans003 这个控件随着温度的变化而变化,在温度小于 40 度时,风扇不转动,温度大于 40 度小于 80 度时温度越高转动速度越快,大于 80 度时以最高转速转动(100),风扇的转速通过 Fans003 这个控件的 MotorSpeed 这个属性控制,为实现这个效果我们还需要实现一个值转换器。类似 T2ColorConverter 文件的创建,创建一个名为 T2MotorSpeed 的文件,添加引用 usingSystem.Windows.Data,添加如下代码:



现在添加 Fans003 控件的 MotorSpeed 属性的数据绑定: 绑定类型选择 ElementName, 绑定到 horizontalLinearGauge 控件的 Value 属性上, 值转换器新 建一个 T2MotorSpeed 并选择这个值转换器。

Create Data Binding for	[Fans003].MotorSpeed	studio			٢
Binding Type: Element	Name				•
Element name		Prope	rties:	Only display matching typ	es
▲ [Window]			ToolTip : (Object)		•
▲ [Grid]			TouchesCaptured : (IEr	numerable <touchdevice>)</touchdevice>	
[Fans003]			TouchesCapturedWithi	in : (IEnumerable <touchdevice>)</touchdevice>	
[VesselTank]			TouchesDirectlyOver :	(IEnumerable <touchdevice>)</touchdevice>	
[RacingGaug	e]		TouchesOver : (IEnume	erable <touchdevice>)</touchdevice>	
horizontalLin	earGauge	•	Triggers : (TriggerBase))	
			Uid : (String)		
			UseLavoutRounding : ((Boolean)	
		Value : (Double)			
		VerticalAlignment : (VerticalAlignment)			
			VerticalContentAlignm	ent : (VerticalAlignment)	
			Visibility : (Visibility)		=
			Width : (Double)		-
		•		•	
		Path:	Value		
 Fewer settings 					
Converter:	T2MotorSpeed •	Bin Bin	dsDirectlyToSource		
Converter Parameter:		Is As	sync		
Binding Mode:	Default	Not	tifyOnSourceUpdated		
UndateSourceTrigger	Default	NotifyOnTargetUpdated			
opdatesourcerngger:		Not	tifyOnValidationError		
String Format:	[] T	Val	idatesOnDataErrors		

至此,我们已经完成了界面的基本控件属性和实际采集数据的绑定,点击运行即可看到实时监控效果。

	MainWindow	×
6 9 ()		

在实际情况下,我们会遇到检测相同模型的若干个实例,比如我们现在已经 创建的 Car1 和 Car2,对于这种情况,实际的模型一般只需要创建一套显示控件 即可(如此例子中简单的四个控件),用这一套控件切换着去显示不同实例的实 时监控数据,切换的实现可以有很多种(这里中按钮实现),这种可以切换着去 显示不同实例数据的数据绑定类型和至此本例的所采用的数据绑定类型有所不 同,至此本例中控件属性和 OPCUA 模型实例的属性数据绑定方式为 OPCUA 绝对 绑定,切换显示不同实例监控数据的实现是依赖于 OPCUA 相对绑定。下文详细 讲述。

从工具窗口拖入两个 RadButton 控件,将其 Content 属性改为 Car1 和 Car2,将其 Name 属性改为 btnCar1 和 btnCar2。将大纲中的 Grid 控件的 Name 属性改为 LayoutRoot。

现在需要修改 HorizontalLinearGauge, VesselTank, RacingGauge 这三个控件的 Value 属性的数据绑定类型,先修改 VesselTank 的 Value 属性的数据绑定,如下图:数据绑定类型选择 OPCUA 相对绑定,然后应首先配置数据上下文(如果已经配置过就不需要再次配置),即图中的 Set Binding Context 按钮点击弹出的窗口配置如下图,

图中左窗口中选中的 LayoutRoot 是包含这一套控件的父级容器,右窗口选中 我们已经建好的 DataHub 中的 TransportCar 这个类型,点击 OK。配置完 Set Binding Context 后点击刷新选择 MaterialLevel 属性的 Value 属性进行绑定,点击 OK。



_ M----

Set BindingContext	
Target Element: 🔲 Set As Design Time Binding Context	Connection Name: AicTech_DataHub_Server Refresh New
▲ [Window]	A 🗁 Root 🔶
LayoutRoot	Dijects
- [Fans003]	🔺 🗁 Types
[VesselTank]	DataTypes
[RacingGauge]	EventTypes
horizontalLinearGauge	ObjectTypes
e radButton	BaseObjectType
radButton1	AggregateConfigurationType
n i	AggregateFunctionType
n	BaseConditionClassType
n	BaseEventType
n	CustomBaseObjectType
ri L	TransportCar
ri	🖾 DataTypeEncodingType
	🖾 DataTypeSystemType
	FileType
	SolderType
	HistoricalDataConfigurationType
	HistoryServerCapabilitiesType
	EcckType
	ModellingRuleType
•	NamespaceMetadataType
Selected Element: LayoutRoot	Node Path: //ypes/Object/ypes/BaseObject/ype/2:CustomBaseC

同样我们完成 HorizontalLinearGauge 控件和 RacingGauge 控件的数据绑定,分别 绑定到 Temperature 的 Value 属性和 Speed 的 Value 属性。至此,真正的数据还 没法到达控件的属性上,参考 4.3.1 中为 btnCar1 和 btnCar2 两个按钮的 Click 事 件分别添加 btnCar1_Click 和 btnCar2_Click 处理方法,参考 5.2 中分别在插入 Set Relative Binding Context 对应的代码段,配置如下图,点击刷新按钮,在 btnCar1_Click 中选择 Car1 对象,在 btnCar2_Click 中选择 Car2 对象。参考 4.2 中的 OPCUA 相对绑 定,将 SetBindingContext 方法中的 this 参数改为 this.LayoutRoot。

Select Node			_ 0 X
Connection Name:	AicTech_DataHub_Server 🔹	Refresh	New
🔺 🗁 Root			
🔺 🗁 Objects			
🕨 🔶 Server			
🔺 🗁 Syster	1		-
🕨 📄 Ala	rms		
🕨 🔶 🍕 Glo	balMethods		
🔺 🗁 Pla	ntObjects		
 • 	Car1		
► 🔩	Car2		, ,
🕨 🚞 Scr	ipts		
🕨 🚞 See	urity		
Types			
🗀 Views			

至此,已经实现了 OPCUA 的相对绑定,运行点击 Car1 和 Car2 按钮就可以进行 Car1 实例和 Car2 实例监测数据的切换。