DirectUI 使用指南

V1.2

上海勇进软件有限公司

目　录

使用指南概要

前言

微软公司从 Windows98、Windows2000 再到今天已经普及的 WindowsXP、Windows7，每次推出新的操作系统都让人眼前一亮，Windows7 更是给我们带来了很长一段时间的视觉冲击波。使得软件界面越来越受到软件开放厂商的重视，然而设计与开发一款优秀的软件的界面并不是非常容易的，如何能快速高效的完成软件界面的设计与开发呢，通过 DirectUI 界面就可以解决这一问题。本指南详细介绍了 DirectUI 界面库与开发方法，通过本指南的学习可以让我们认识 DirectUI 界面库，掌握 DirectUI Builder 的使用方法和 DirectUI 与程序集成开发的相关技术。

面向的读者

本指南只要面向准备学习 DirectUI 开发技术的界面开发人员，使用 DirectUI Builder 制作皮肤的皮肤制作人员，以及想要了解 DirectUI 开发技术的开发人员。对于第一次接触 DirectUI 的开发人员建议先通过“DirectUI 开发实例”掌握 DirectUI 界面库的基本使用方法。

本书主要内容

第一章，DirectUI 产品概述——介绍了 DirectUI 产品构架及特性。
第二章，DirectUI 界面开发流程——介绍了使用 DirectUI 界面库进行界面程序开发的流程。
第三章，利用 DirectUI 进行产品开始实例——通过一个例子了解基于 DirectUI 界面库的客户端产品开发流程。
第四章，DirectUI 界面设计规范——介绍了 DirectUI 界面控件皮肤切图的规格和制作要求。
第五章，DirectUI Builder 皮肤制作介绍——介绍了皮肤制作工具 DirectUI Builder 的使用方法并介绍控件属性的含义。
第六章，DirectUI 界面库的调用方法——介绍 DirectUI 界面库在开发工具中如何进行调用
第七章，利用 JavaScript 控件 DirectUI 控件——介绍如何编写 JavaScript 脚本来控件 DirectUI 控件以及 DirectUI 控件如何响应脚本代码。
第八章，DirectUI 多语种界面开发——介绍如何使用 DirectUI Builder 创建新的界面语言及切换语言的方法。
第九章，DirectUI 多皮肤开发——介绍如何使用 DirectUI Builder 创建新的皮肤及皮肤动态切换方法。
第十章，DirectUI 界面库扩张方法——介绍如何开发 DirectUI 控件，并介绍 Button 按钮的开发实例。
第十一章，DirectUI 使用 FAQ——介绍常见的 DirectUI 使用问题的解决方法。

一、 DirectUI 产品概述

1. DirectUI 产品介绍

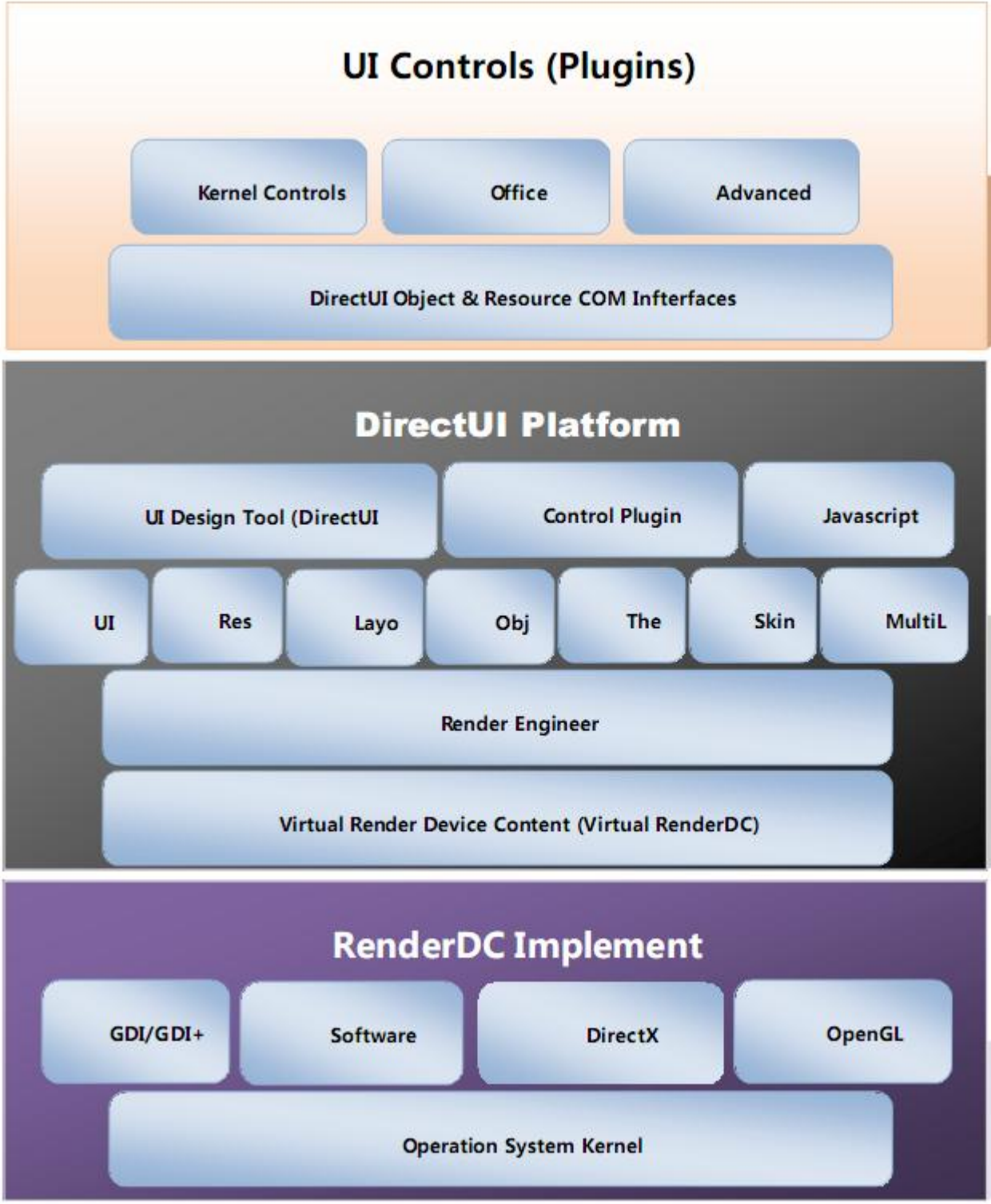
DirectUI 为上海勇进公司于 2005 年开发的一款软件界面开发工具，可以快速高效的完成软件界面的开发工作。DirectUI 界面库使用 XML 来描述界面风格，界面布局，使用脚本语言（JavaScript）来实现界面逻辑与业务逻辑的彻底分离。

2. DirectUI 产品特性

1) 支持皮肤对象的布局；
2) 支持皮肤脚本(JavaScript)控制,可以让界面与逻辑彻底分离；
3) 支持 bmp、png、jpg、gif、tga 等图片格式；
4) 支持多图层 Alpha 混合特效；
5) 提供界面设计工具 DirectUI Builder,支持拖拽式界面设计，让界面开发所见即所得；
6) 开放式开发平台，所有控件均为插件方式管理，支持用户自定义控件开发，与 DirectUI 平台无缝兼容；
7) 支持 Windows 平台所有的开发工具(VC++、VB6、VS.Net、PowerBuilder、Delphi、C++Builder、E 语言)；
8) 支持所有标准控件的换肤；
9) 支持皮肤对象的导出不导入；
10) 支持 Windows 主题导入，让标准界面皮肤的制作简单快捷；

3. DirectUI 产品构架

DirectUI 采用平台加插件的模式，平台提供界面库运行环境支持，插件为界面库中使用到的各类控件。



- **DirectUI 采用平台+插件的体系架构**。DirectUI 2 系列版本中所有的控件均为插件方式管理。常用的控件被封装在 KernelAll 控件工程中。用户可以使用插件开发向导来开发自己的控件。在 DirectUI 以后的版本中，插件将分几种类型：控件插件、特效插件、图像解码插件、脚本插件等。
- **DirectUI 平台包含：**绘图引擎、UI 逻辑、事件处理、对象布局、脚本控制、共享资源、插件控制器、标准控件换肤引擎、多语种控制器、界面开发可视化工具 DirectUIBuilder 等。
- **DirectUI 的绘图引擎**是最核心的功能模块，其功能强弱、效率高低、内存占用高低等都直接影响到 DirectUI 整体的性能。绘图引擎采用纯虚的图像设备上下文来对各种操作系统进行全面的支持。Virtual RenderDC 将各种图像与文字的处理设计成各种类，并将每个类的方法设计成纯虚函数。如果要支持例如 DirectX 绘图引擎，只需要将那些纯虚的各种类与接口实现即可。
- **UI Logic：**负责处理所有控件共用的各种逻辑。

- **Share Resource**：DirectUI 将所有的图片、字体、图像选区、文本样式、字体、光标等称之为资源，并将这些资源进行计数引用式管理，从而很好地避免数据的冗余与内存的最小化。
- **Object Layout Manager :**DirectUI 中所有的对象均有布局的概念，并且布局器支持 2 种以上的布局：父子布局、表格布局、绝对布局等。有了该项功能后用户不再需要程序中对各种控件进行布局了。对象布局信息被包含在皮肤方案文件中，所以更换一套皮肤方案就能更改一种布局。
- **Control Objects**：DirectUI 中将所有的控件统称为对象，在平台层将每个控件共用的属性与行为进行了基类的封装。
- **DirectUI Builder :** DirectUI 提供了所见即所得的界面开发工具 DirectUIBuilder。用户通过该工具可以对界面进行可视化的设计：支持拖拽式的控件布局，支持控件的拷贝与粘贴，支持控件的导出与导入，对大团队开发进行了强有力的支持。
- **Control Plugin Manager :** DirectUI 中所有的控件均为插件。在平台层有对所有类型插件进行管理的机制，方便用户编写各种类型的插件，提供控件开发的控件型插件向导。用户通过该向导可以一步一步地将控件与平台接口的各种属性进行确定，最终生成一个可供 DirectUI Builder 无缝兼容的新控件。
- **Javascript Control :** DirectUI 通过 Javascript 的控制来真正实现界面逻辑与业务逻辑的彻底分离。Javascript 语法简单很容易被 C++ 程序员掌握，DirectUI 采用 COM 技术对 Javascript 的功能对象进行了扩充。将 DirectUI 中所有的控件接口自动添加进 Javascript 引擎，从而可以在 Javascript 中方便地访问各种控件接口中的方法。脚本功能被包含在 DirectUIBuilder 中，用户填写的脚本代码被包含在皮肤方案文件 SKN 中，所以更换了 SKN 文件，脚本的界面逻辑控制代码也随即一起更换了。
- **Theme :** DirectUI 支持多主题功能，通过主题用户可以节约大部分的控件属性的重复设置工作。一般的软件界面项目中 80%的控件均为相同样式的，只有很少的大概占 20%的控件才需要特殊的风格设置，所以主题是解决该 80%控件重复设置的问题，同时主题还可以跨项目跨软件使用，其重用程度相当于 Windows 的主题文件。
- **SkinCtrl**： 在一套软件界面中，从总体上来说分标准控件的换肤与自定义控件界面定义。SkinCtrl 就用来对 Windows 标准控件进行换肤的模块。即使在用户的程序中没有用到一个 Windows 标准控件，但还是需要对 SkinCtrl 模块的界面属性进行设置。因为在所有的程序中几乎都有 MessageBox、打开文件对话框、颜色选择对话框、目录选择对话框等 Windows 系统窗口，这些窗口都属于标准控件的范畴。
- **MultiLang:** DirectUI 支持多语种功能，用户只要使用 DirectUIBuilder 中多语种管理窗口即可实现多国语种的管理与动态切换语种的功能。
- **UI Controls(Plugins) :** 目前 2.0 版本中控件分成 4 大系列：核心控件组、办公类软件控件组、高级类控件组、工业类软件控件组。这些控件都是以 DirectUI 的插件形式存在，可以单独加载与卸载。给程序发布提供了灵活的解决方案。

二、 DirectUI 界面开发流程

1. DirectUI 界面开发流程介绍

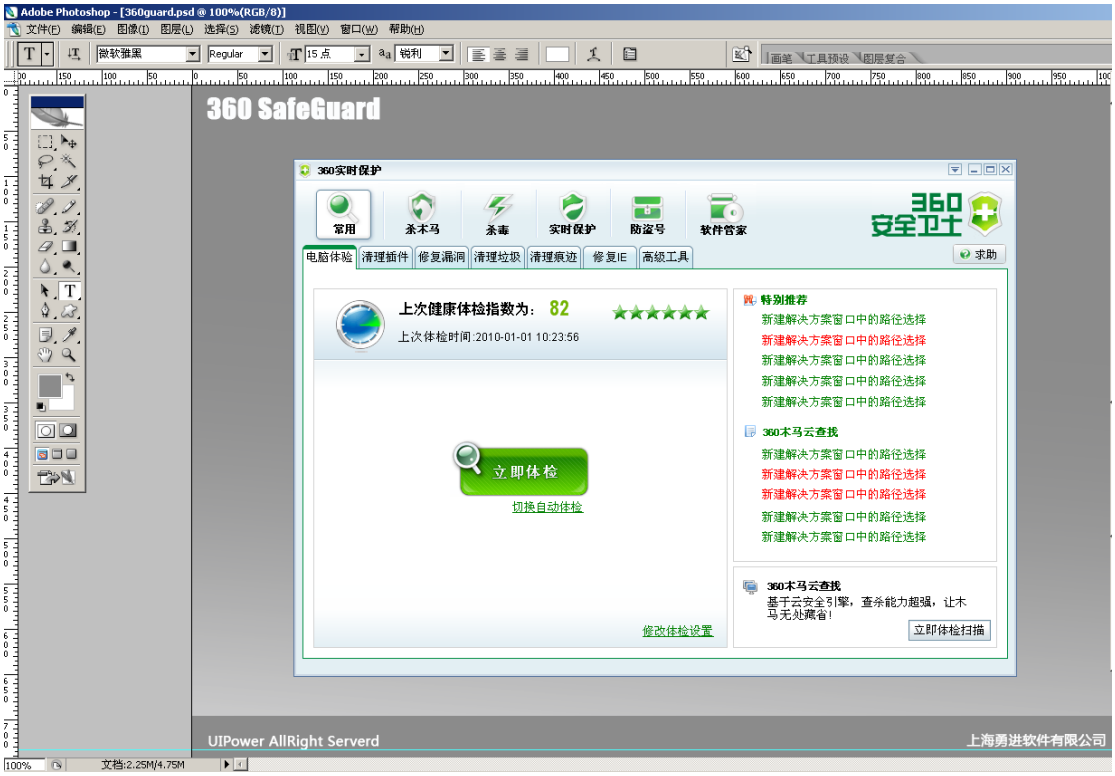
1） 使用 PhotoShop 进行产品的交互及视觉设计完成产品最终效果图；
2） 使用 PhotoShop 对效果图进行切图提取其中的控件切图；
3） 使用 DirectUI Builder 制作界面皮肤；
4） 使用软件开发工具如 Visual Studio 开发产品的界面模块

三、 利用 DirectUI 进行产品开发实例

我们将利用 DirectUI 界面库演示制作一个 360 杀毒软件的界面 Demo。

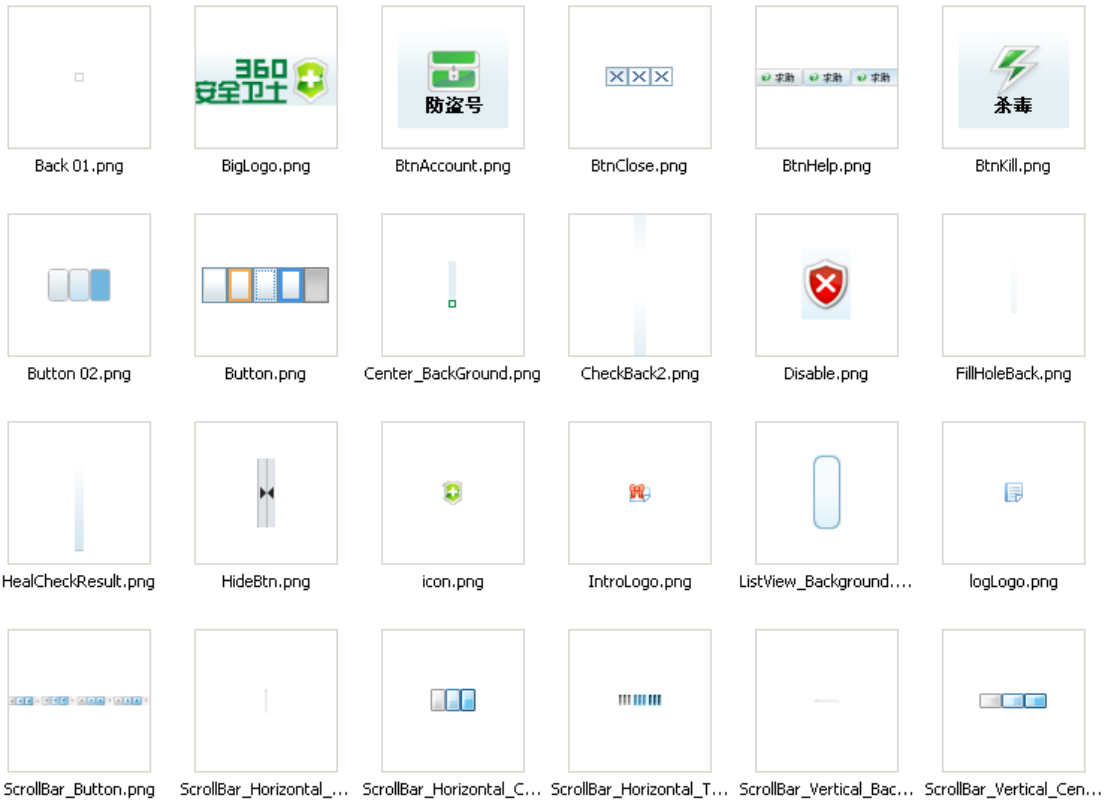
1. 使用 PhotoShop 设计软件效果图

启动 Photoshop，对 360 杀毒软件的界面进行设计和绘制。



2. 利用 Photoshop 对效果图进行切图

将各个控件的图层进行处理，分别保存为 PNG 文件。



3. 使用 DirectUI Builder 制作界面皮肤

3.1 Spp 文件转换

由于 DirectUI Builder 只能识别 SPP 图片，所以我们需要将 PNG 图片装换为 Direct Builder 可识别的 SPP 图片。我们将使用 UIPower 的 SPP 在线 SPP 转换工具将 png 图片进行转换。

a) 首先将切图 png 文件打包为一个 zip 文件。

需要注意的是图片处于 zip 包的根目录下，非根目录下的图片将 不会被转换。

b) 打开浏览器，输入网址 http://www.uipower.com/spp，打开后将出现 SPP 在线转换工具的登录界面。输入用户名，密码将可以登录到转换页面。

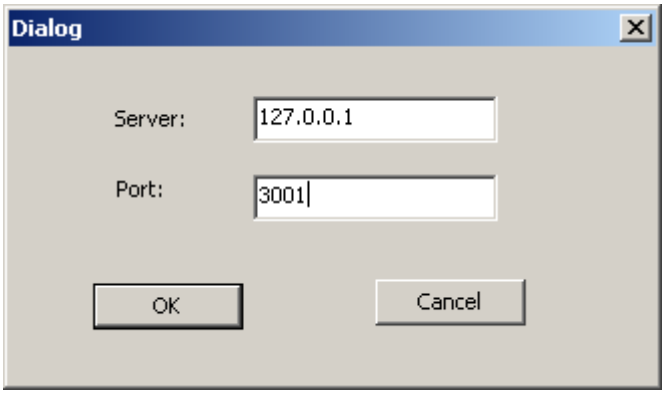
用户名、密码请联系上海勇进软件公司索取，联系电话 021-34021068。

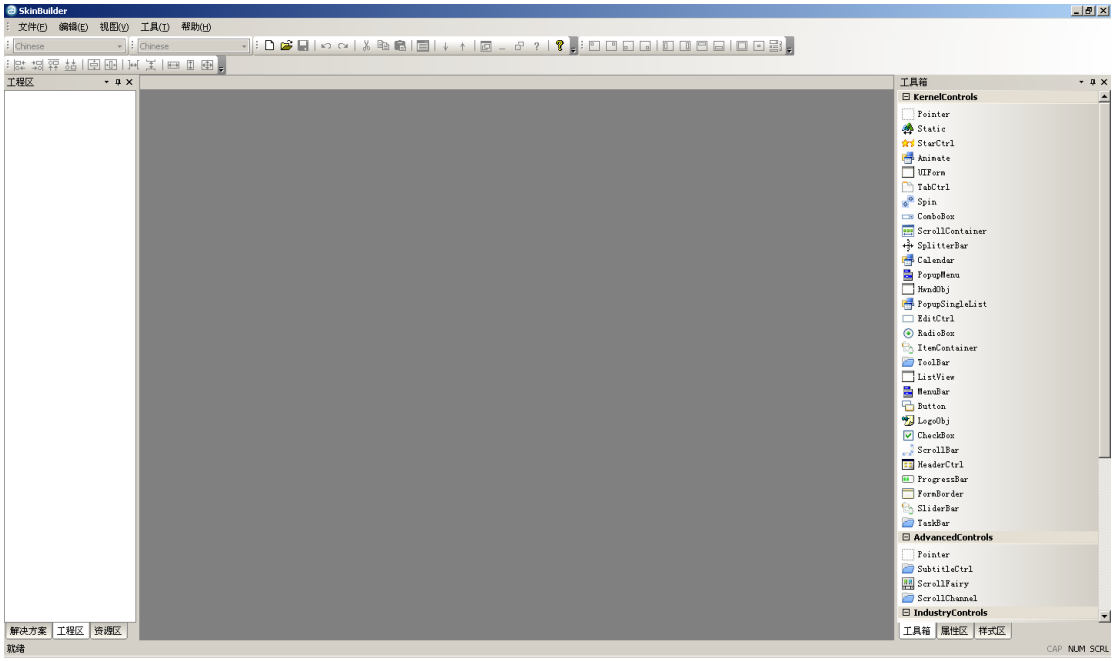
c) 选择之前我们压缩的 zip 包，点击“转换”按钮，服务器会将图片转换为 SPP 后重新压缩为一个 zip 包，下载此压缩包即可获取转换完成的 spp 文件。

3.2 启动 DirectUIBuilder

1） 在本地插入硬件狗，启动 DirectUI Builder

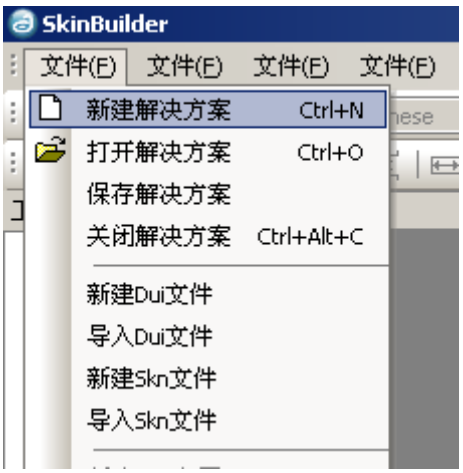
2）弹出许可管理器配置窗口，填写服务器为 127.0.0.1，端口为 3001，启动 DirectUI Builder



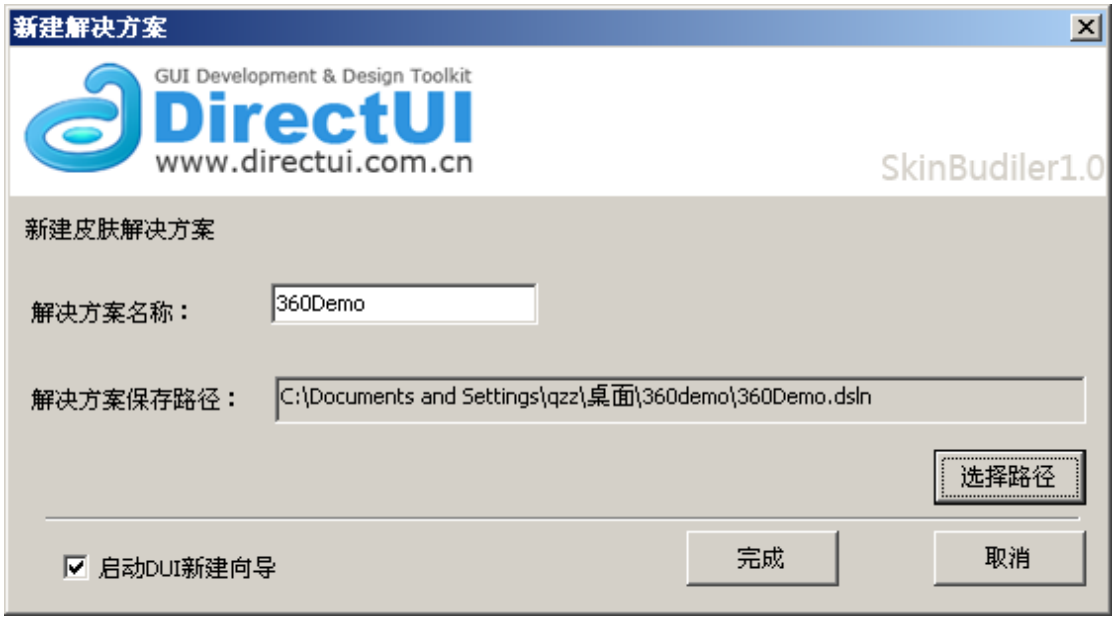


新建解决方案

1）选择文件，新建解决方案



弹出解决方案配置窗口，输入解决方案的名称并选择保存的路径。



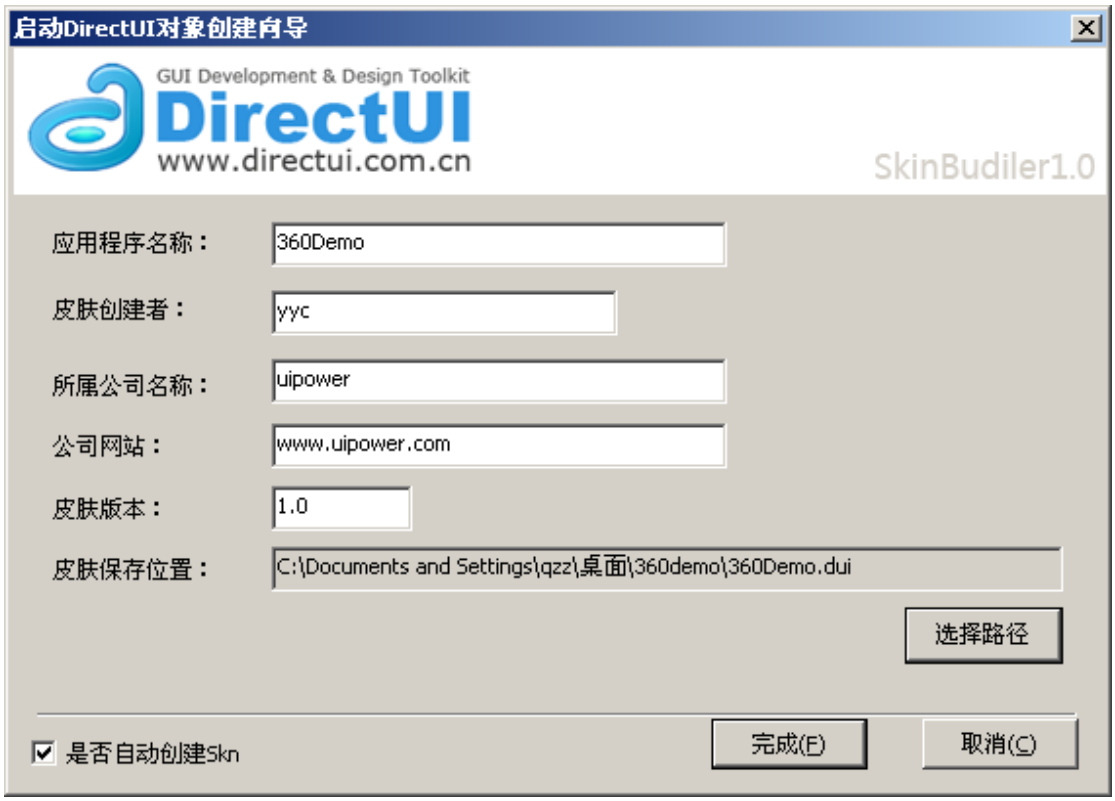
我们这里解决方案的名称为 360Demo，保存路径为桌面上 360Demo 文件夹。

将“启动 DUI 新建向导”处于勾取状态。

点击“完成”按钮，进入 DUI 文件的窗口步骤。

2）完成后将可以看到 DUI 文件的创建窗口，分别填写皮肤信息和 DUI 文件保存路径。

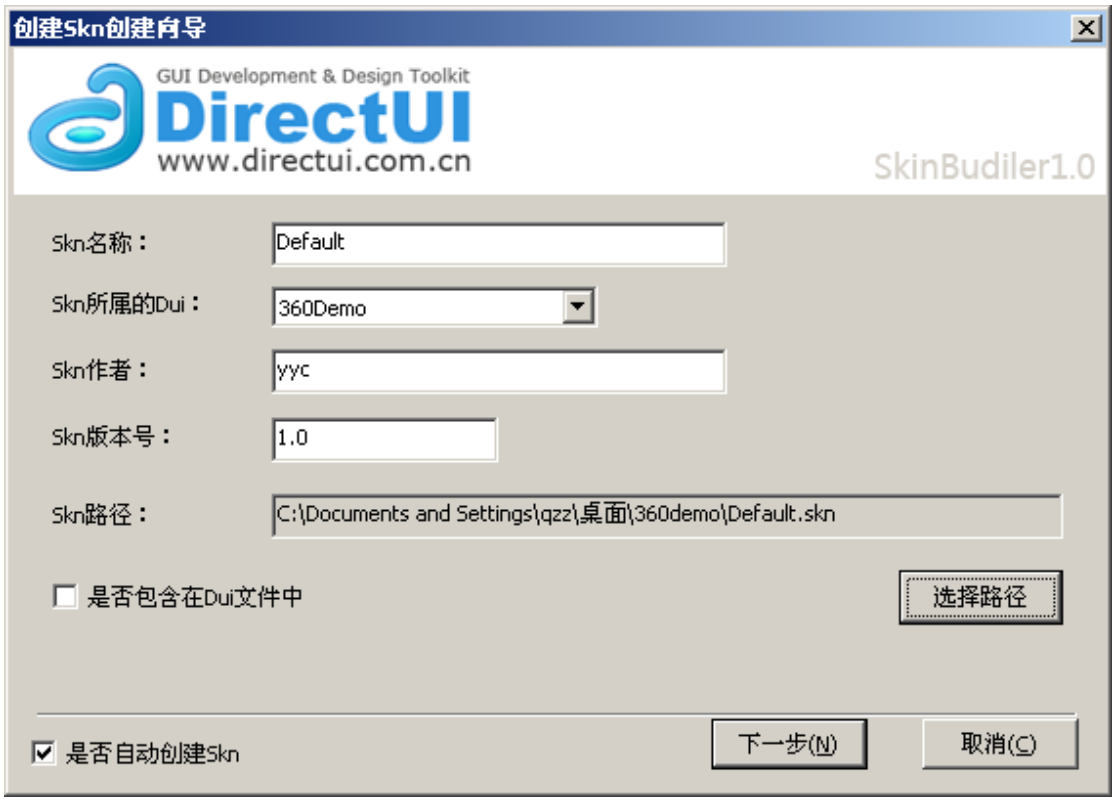
DUI 文件描述了界面窗口的结构关系，是皮肤骨架的描述文件。



将“是否自动创建 Skn”选择为勾取状态，点击“完成”按钮，进入 Skn 文件配置窗口。

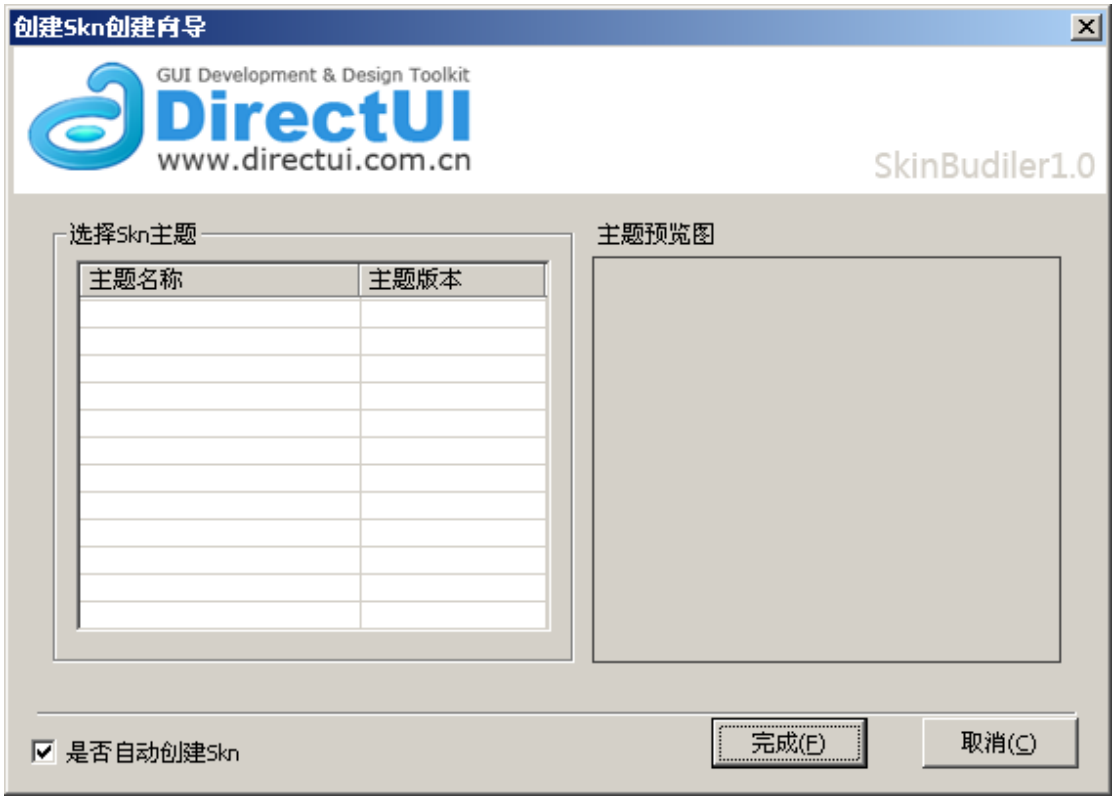
3）在 Skn 文件配置窗口中填写 Skn 配置信息和保存路径后点击“下一步”按钮。

Skn 文件描述了界面控件的外观，是窗口的外皮。



4）点击“下一步”后进入皮肤主题的选择窗口

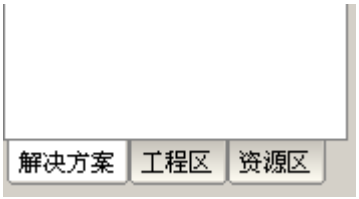
由于目前我们未发布任何的主题包，所以没有主题可以进行选择，直接点击“完成”按钮。

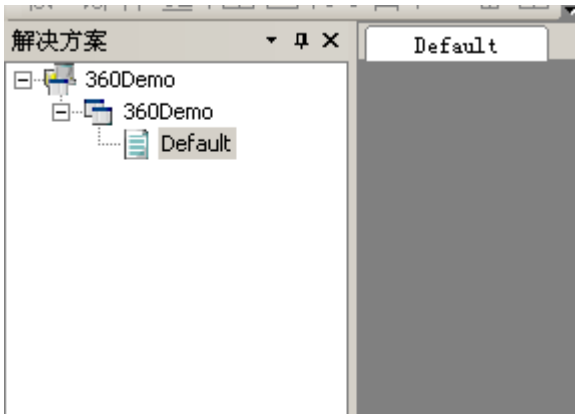


进过以上的步骤我们完成了皮肤文件的创建工作，接下来我们将开始设置皮肤控件。

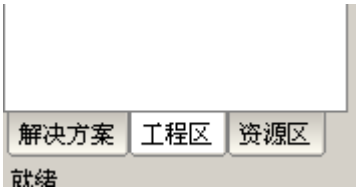
设置皮肤控件

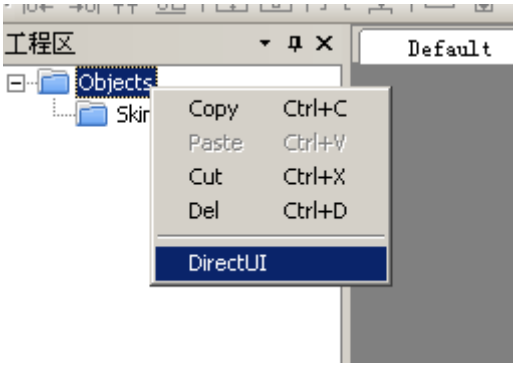
1）选择解决面板，双击 Default，打开 Default 皮肤，在工程预览区我们将会看到 Default 皮肤的标签。



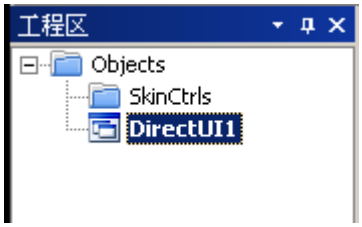


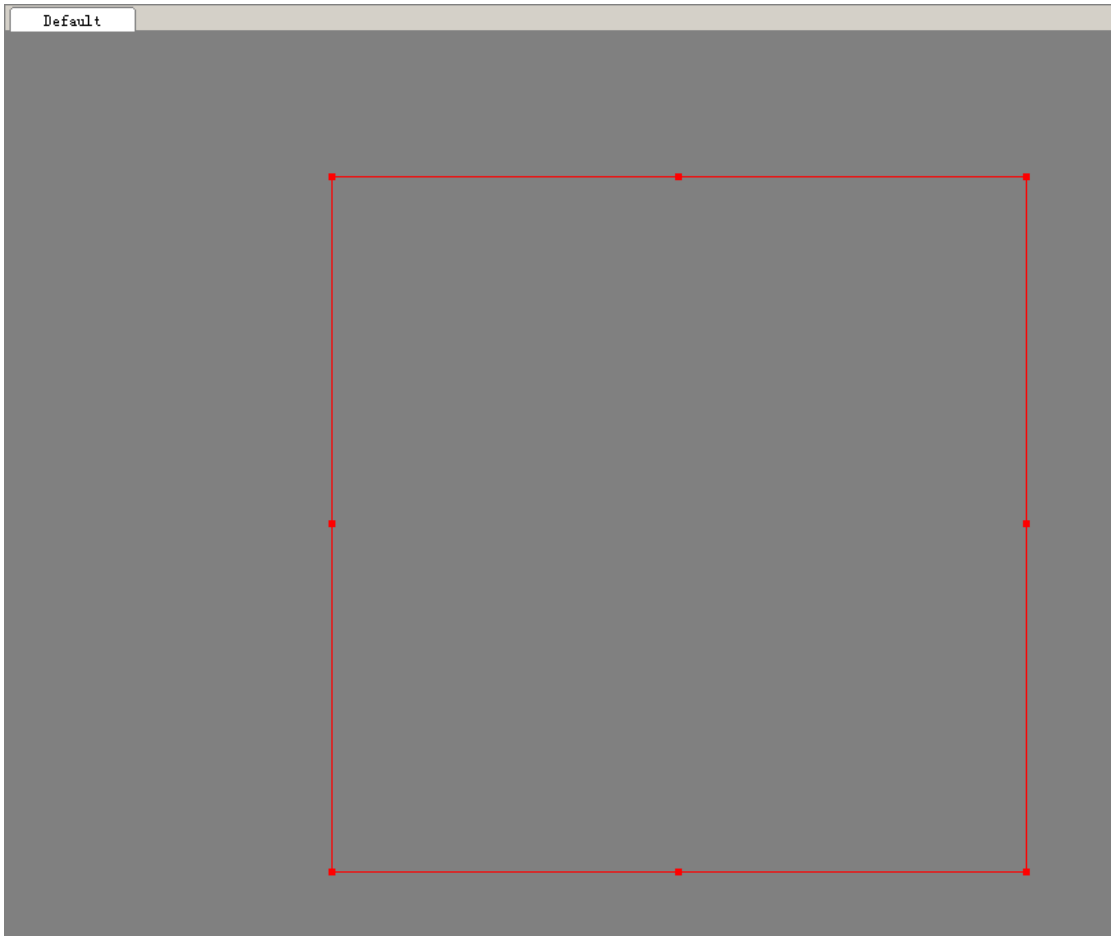
2）选择工程区，我们打开工程面板，在 Object 节点上我们单击右键，选择“DirectUI”



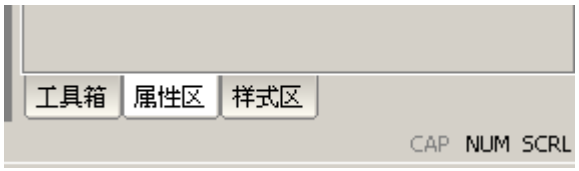


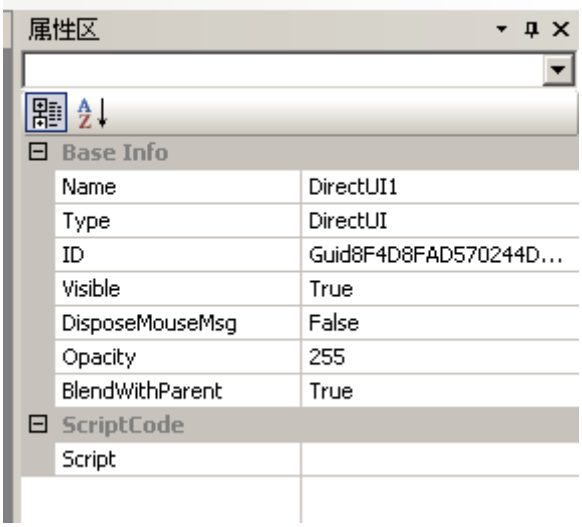
我们会建立一个 DirectUI 窗口，选择此点击后工程预览区将出现一个红框，表示此控件的范围。



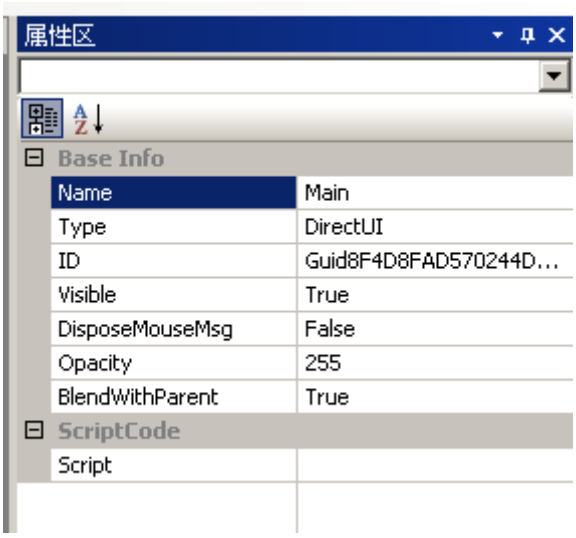


3）切换到属性区，我们将对此 DirectUI 进行属性设置。





4）修改“Name”为“Main”

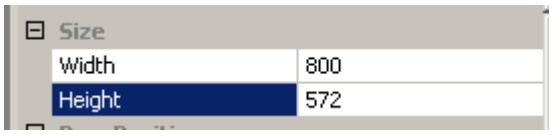


5）切换到演示区设置此窗口的外观。

设置一个启动默认大小

Width：800

Height：572



窗口可拖拽

Dragable：True

窗口最小范围 800*572

MinWidth：800

MinHeight：572

弹出窗口类型

Popwidow：True

根据图片计算 RGN 区域

SetRgn：True

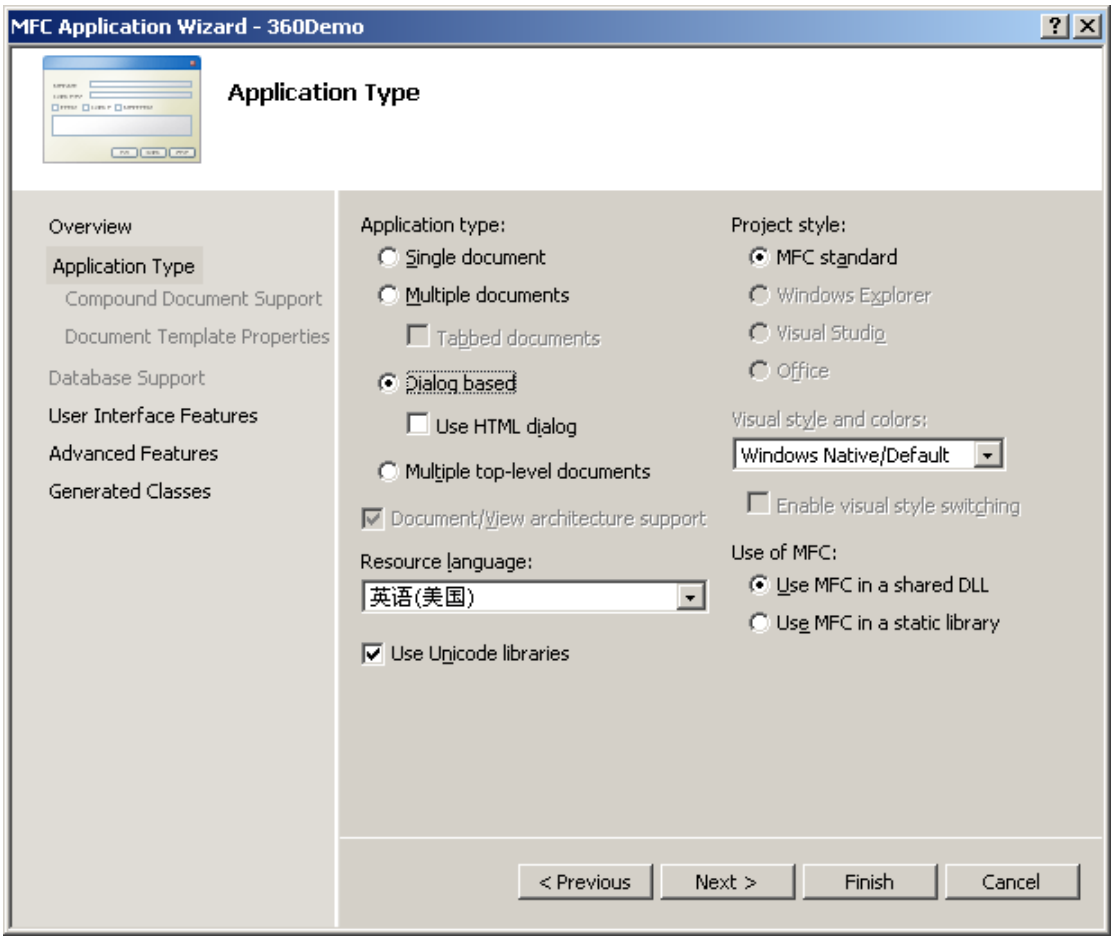
Misc	
Dragabled	True
DragFullWindow	True
CanCopyRun	False
MinWidth	800
MinHeight	572
MaxWidth	-1
MaxHeight	-1
TopMost	False
MaxToScreenAll	False
SupportPerPixel	False
PopWindow	True
ParentBlend	False
SetRgn	True
UseRgnInfo	False
SysMenu	True
MinBox	True
MaxBox	False

6）其他的控件设置参考 DirectUI 使用视频教程。

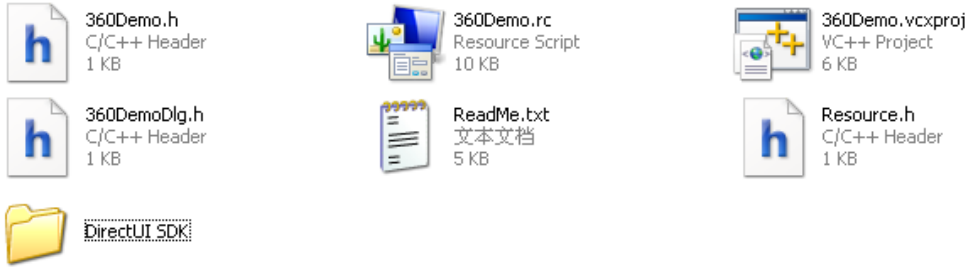
4. 使用 VisualStudio2010 建立工程完成界面模块的开发

4.1 新建工程

1） 打开 VS2010，新建一个 MFC 对话框工程，工程名称为 360Demo



2） 打开之前创建的工程文件目录，将 DirectUI SDK 文件夹放置到工程目录下。



3） 将 DirectUI 动态库拷贝到工程目录下

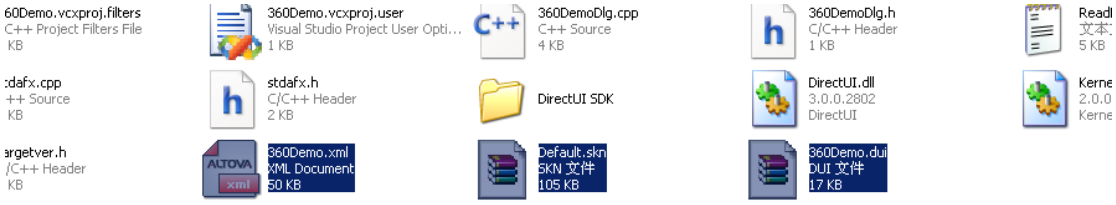
DirectUI.dll

KernelAll.dll

DirectUI.lib

由于我们 360Demo 仅仅使用 KernelAll.dll 中的控件，所以我们仅仅包含了 DirectUI 平台文件 DirectUI.dll，KernelAll 控件文件 KernelAll.dll，而没有拷贝 OfficeAll.dll、AdvancedAll.dll 等其他控件动态库。

4） 拷贝 360 的皮肤文件到程序目录中。

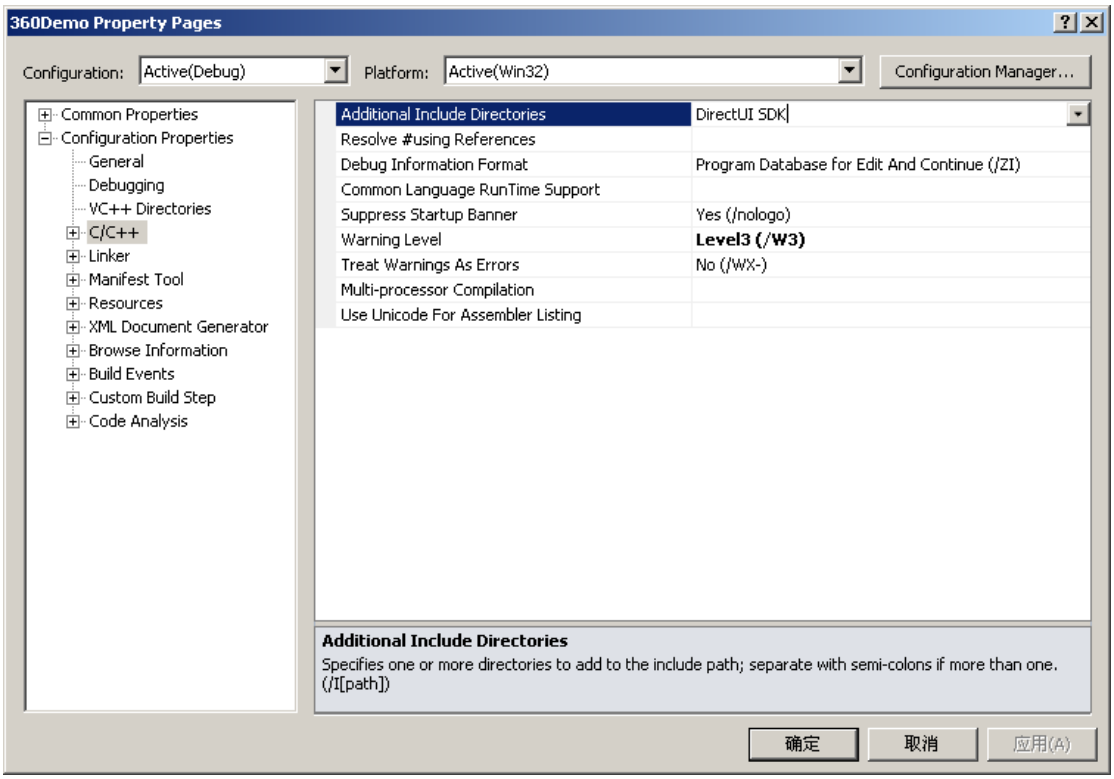


皮肤框架文件 360Demo.dui

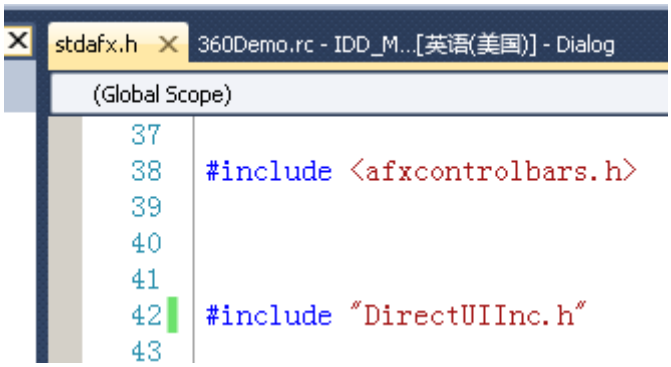
皮肤外观文件 Default.skn

界面语言配置文件 360Demo.xml

5） 设置工具属性，将 DirectUI SDK 包含到 Include 目录中。

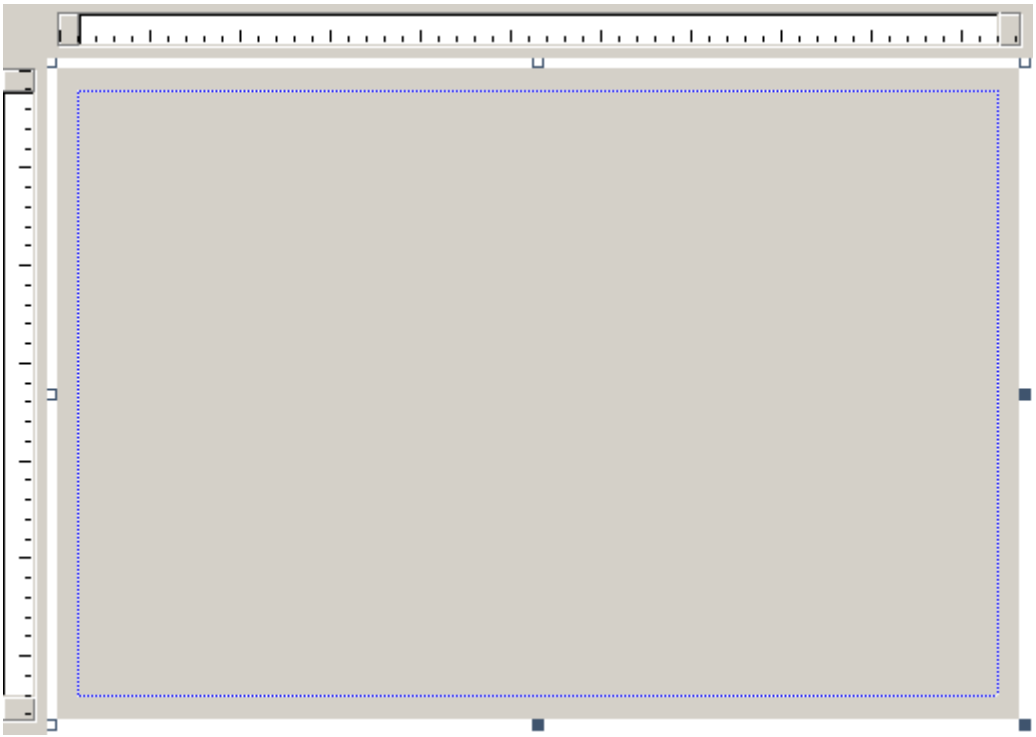


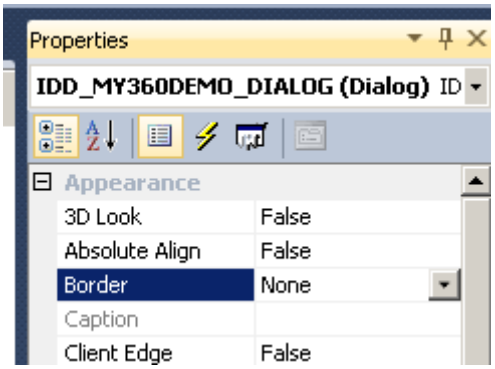
6） 打开 Stdafx.h 文件，将 DirectUIInc.h 文件包含到工程中。



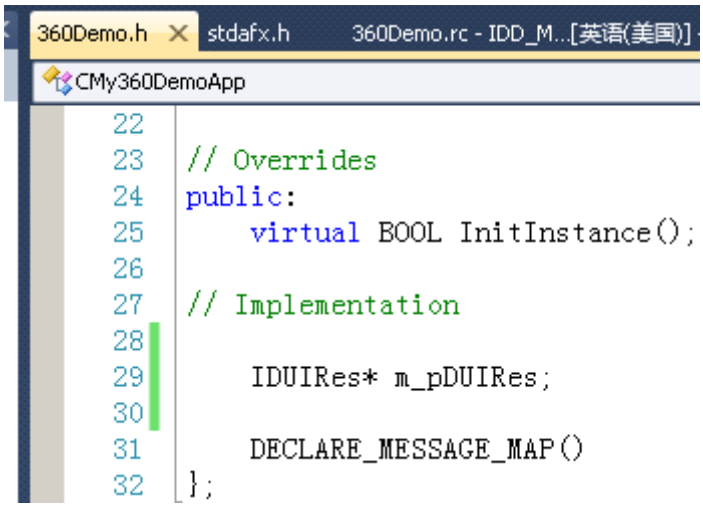
7） 打开资源窗口中的主窗口，将窗口上的控件全部删除

并设置窗口样式中 Border 为 None





8） 打开 360Demo.h 文件，在 APP 类中添加 DirectUI 管理对象 IDUIRes 指针



在程序 InitInstance 方法中，调用 IDUIRes 的 OpenSkin 方法，打开 DirectUI 皮肤，OpenSkin 参数分别填写 DUI 和 SKN 文件名称。

```
360Demo.cpp  360Demo.h  stdafx.h  360Demo.rc - IDD_M...[英语(美国)] - Dialog
CMy360DemoApp                                   InitInstance()
49      InitCtrls.dwICC = ICC_WIN95_CLASSES;
50      InitCommonControlsEx(&InitCtrls);
51
52      CWinApp::InitInstance();
53
54      m_pDUIRes = OpenSkin(_T("360Demo.dui"), _T("Default.skn"));
55
56
```

在 ExitInstance 方法中调用 FreeSkin 方法。

```
360Demo.cpp  360Demo.h  stdafx.h  360Demo.rc - IDD_M..
CMy360DemoApp
97
98
99   int CMy360DemoApp::ExitInstance()
100  {
101      // TODO: Add your specialized co
102
103      FreeSkin();
104
105      return CWinApp::ExitInstance();
106  }
```

9） 打开主窗口头文件，声明一个 DirectUI 对象 IDirectUI。

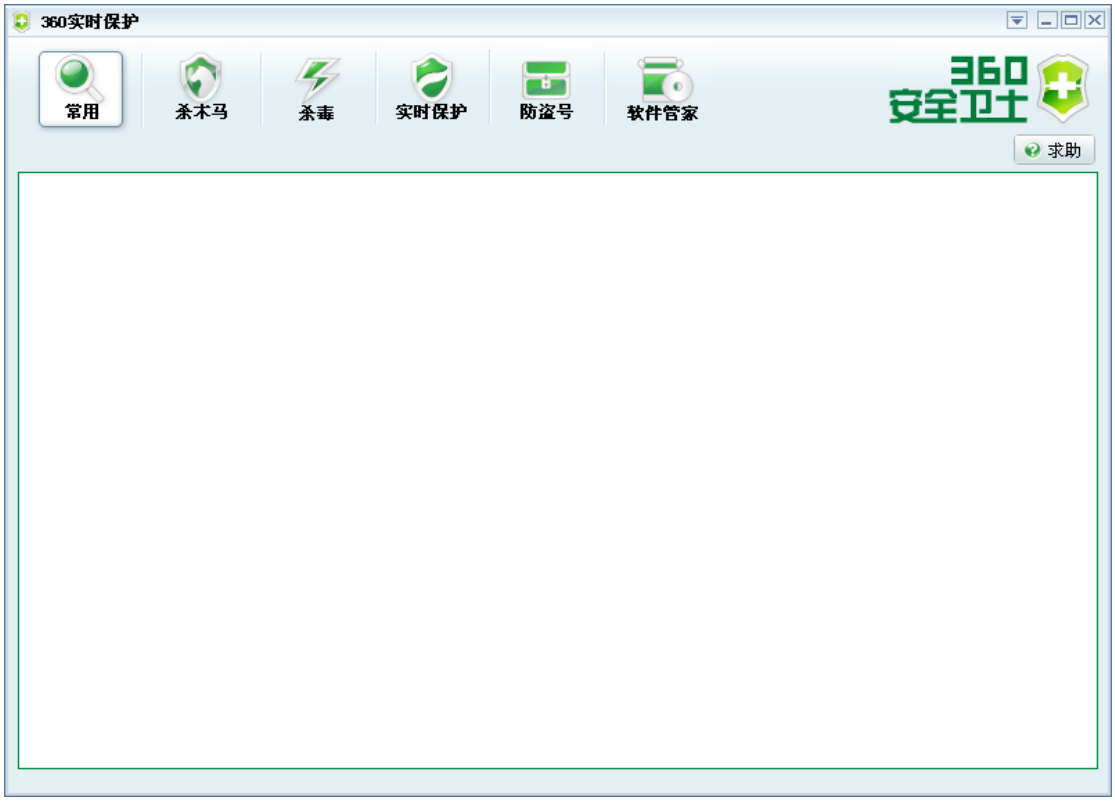
```
360DemoDlg.h  360DemoDlg.cpp  360Demo.cpp  360
CMy360DemoDlg
25
26      // Generated message map fur
27      virtual BOOL OnInitDialog();
28      afx_msg void OnSysCommand(UI
29      afx_msg void OnPaint();
30      afx_msg HCURSOR OnQueryDragI
31      DECLARE_MESSAGE_MAP()
32
33  private:
34      IDirectUI* m_pDirectUI;
35  };
36
```

在主窗口中的 OnCreate 方法中调用 IDUIRes 的 CreateDirectUI 方法获取 DirectUI 对象，并将此窗口的句柄绑定到此 DirectUI 对象上。

```
360DemoDlg.h  360DemoDlg.cpp  360Demo.cpp  360Demo.h  stdafx.h  360Demo.rc - IDD_M...[英语(美国)] - Dialog
CMy360DemoDlg                                   OnCreate(LPCREATESTRUCT lpCreateStruct)
154
155
156 int CMy360DemoDlg::OnCreate(LPCREATESTRUCT lpCreateStruct)
157 {
158     if (CDialogEx::OnCreate(lpCreateStruct) == -1)
159         return -1;
160
161     // TODO:  Add your specialized creation code here
162     m_pDirectUI = (IDirectUI*)theApp.m_pDUIRes->CreateDirectUI(_T("Main"), (long)HandleToLong(GetSafeHwnd()));
163
```

m_pDirectUI = (IDirectUI*)theApp.m_pDUIRes->CreateDirectUI(_T("Main"), (long)HandleToLong(GetSafeHwnd()));

10） 编译调试将看到 360 的主窗口被调用出来。

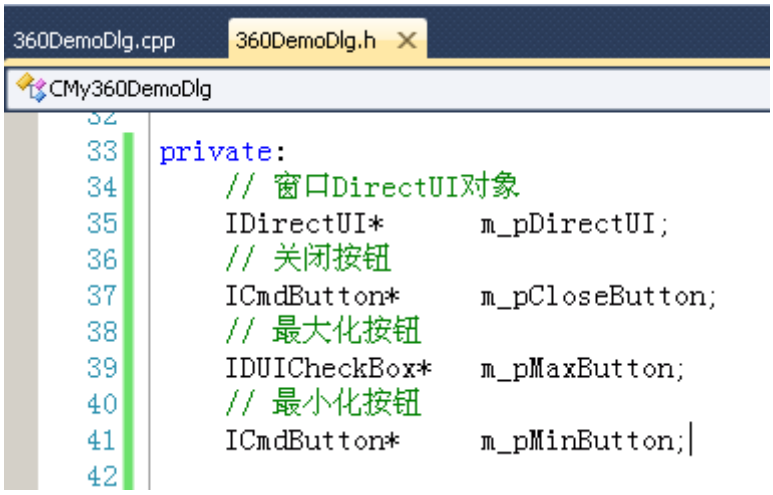


4.2 DirectUI 控件的获取

与 mfc 的控件相同，操作 DirectUI 窗口需要先获取窗口上的控件，下面我们分别获取右上角的系统按钮控件，最小化、最大化、关闭按钮。

1） 在主窗口的头文件中申明三个按钮控件

```
ICmdButton*      m_pCloseButton;
ICheckBox*       m_pMaxButton;
ICmdButton*      m_pCloseButton;
```



2） 在窗口的 OnCreate 方法中我们分别获取这几个控件

这里我们通过 DirectUI 对象的 GetObjectByCaption 方法来获取这个 DirectUI 对象中的控件。GetObjectByCaption 是通过控件在皮肤中设置的名称来获取这个控件。

如关闭按钮在皮肤中设置的名称为 BtnClose。

```
m_pCloseButton = (ICmdButton*)m_pDirectUI->GetObjectByCaption(DUIOBJTYPE_PLUGIN, _T("BtnClose"),
```

TRUE);
m_pMinButton = (ICmdButton*)m_pDirectUI->GetObjectByCaption(DUIOBJTYPE_PLUGIN, _T("BtnMin"), TRUE);
m_pMaxButton = (IDUICheckBox*)m_pDirectUI->GetObjectByCaption(DUIOBJTYPE_PLUGIN, _T("BtnMax"), TRUE);

```
// 获取关闭按钮
m_pCloseButton = (ICmdButton*)m_pDirectUI->GetObjectByCaption(DUIOBJTYPE_PLUGIN, _T("BtnClose"), TRUE);
ASSERT(m_pCloseButton);
// 获取最小化按钮
m_pMinButton = (ICmdButton*)m_pDirectUI->GetObjectByCaption(DUIOBJTYPE_PLUGIN, _T("BtnMin"), TRUE);
ASSERT(m_pMinButton);
// 获取还原按钮
m_pMaxButton = (IDUICheckBox*)m_pDirectUI->GetObjectByCaption(DUIOBJTYPE_PLUGIN, _T("BtnMax"), TRUE);
ASSERT(m_pMaxButton);
```

4.3 系统按钮响应

DirectUI 控件事件的响应可以通过 DirectUI 平台抛出的自定义消息或者继承一个事件虚接口的方式了实现，这里我们先通过自定义消息的方式来响应按钮事件。

处理平台部内的自定义消息 DUISM_LBUTTONUP

1） 首先声明一个消息处理方法

afx_msg LRESULT OnDUILButtonUp(WPARAM wparam, LPARAM lparam);

```
// 最小化按钮
ICmdButton*     m_pMinButton;

blic:
afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
afx_msg LRESULT OnDUILButtonUp(WPARAM wparam, LPARAM lparam);
```

2） 映射自定义消息

ON_MESSAGE(DUISM_LBUTTONUP, OnDUILButtonUp)

```
BEGIN_MESSAGE_MAP(CMy360DemoDlg, CDialogEx)
    ON_WM_SYSCOMMAND()
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    ON_WM_CREATE()
    ON_MESSAGE(DUISM_LBUTTONUP, OnDUILButtonUp)
END_MESSAGE_MAP()
```

3） 处理按钮事件

DUISM_LBUTTONUP 中 wparam 传入的是一个控件的指针，所以通过指针对比我们就可以知道我们当前点击的是否是关闭按钮；

```
LRESULT CMy360DemoDlg::OnDUILButtonUp(WPARAM wParam, LPARAM lParam)
{
    IDUIControlBase* pControl = (IDUIControlBase*)wParam;
    if(pControl == NULL)
```

```
    {
        return 0;
    }

    if(pControl == m_pCloseButton)// 关闭按钮
    {
        OnCancel();
    }
    else if(pControl == m_pMinButton) // 最小化按钮
    {
        ShowWindow(SW_MINIMIZE);
    }
    else if(pControl == m_pMaxButton) // 最大化 还原按钮
    {
        DUICHECKBOX_VALUE eValue = m_pMaxButton->GetValue();
        if(eValue == DUICHECKBOX_CHECKED)
        {
            ShowWindow(SW_MAXIMIZE);
        }
        else
        {
            ShowWindow(SW_RESTORE);
        }
    }

    return 0;
}
```

```
DirectUI.h    DirectUI.tli    360DemoDlg.cpp*    360DemoDlg.h
emoDlg                                                  OnDUILE

LRESULT CMy360DemoDlg::OnDUILButtonUp(WPARAM wParam, LPARAM l
{
    IDUIControlBase* pControl = (IDUIControlBase*)wParam;
    if(pControl == NULL)
    {
        return 0;
    }

    if(pControl == m_pCloseButton)  // 关闭按钮
    {
        OnCancel();
    }
    else if(pControl == m_pMinButton) // 最小化按钮
    {
        ShowWindow(SW_MINIMIZE);
    }
    else if(pControl == m_pMaxButton) // 最大化 还原按钮
    {
        DUICHECKBOX_VALUE eValue = m_pMaxButton->GetValue();
        if(eValue == DUICHECKBOX_CHECKED)
        {
            ShowWindow(SW_MAXIMIZE);
        }
        else
        {
            ShowWindow(SW_RESTORE);
        }
    }
    return 0;
```

4.4 Tab 控件的插入

接下来我们声明一个 tab 控件指针，并在 OnCreate 中获取此控件，然后调用其 InsertItem 方法向 tab 中插入新的 item。

```
// TAB控件
m_pTabCtrl = (IDUITabCtrl*)m_pDirectUI->GetObjectByCaption(DUIOBJTYPE_PLUGIN, _T("TabCtrl"), TRUE);
ASSERT(m_pTabCtrl);

m_pTabCtrl->InsertItem(0,L"电脑体验",1,L"电脑体验");
m_pTabCtrl->InsertItem(1,L"清理插件",2,L"清理插件");
m_pTabCtrl->InsertItem(2,L"修复漏洞",3,L"修复漏洞");
m_pTabCtrl->InsertItem(3,L"清理垃圾",4,L"清理垃圾");
m_pTabCtrl->InsertItem(4,L"清理痕迹",5,L"清理痕迹");
m_pTabCtrl->InsertItem(5,L"修复 IE",6,L"修复 IE");
m_pTabCtrl->InsertItem(6,L"高级工具",7,L"高级工具");
m_pTabCtrl->SetSelectedItem(0);
```

```
ctUI.h    DirectUI.tli    360DemoDlg.cpp    360DemoDlg.h
                                  OnCreate(LPCREATESTRUCT lpCreateStruct)
ASSERT(m_pMaxButton);
// TAB控件
m_pTabCtrl = (IDUITabCtrl*)m_pDirectUI->GetObjectByCaption(DUIOBJTYPE_PLUGIN, _T("TabCtrl")
ASSERT(m_pTabCtrl);

m_pTabCtrl->InsertItem(0,L"电脑体验",1,L"电脑体验");
m_pTabCtrl->InsertItem(1,L"清理插件",2,L"清理插件");
m_pTabCtrl->InsertItem(2,L"修复漏洞",3,L"修复漏洞");
m_pTabCtrl->InsertItem(3,L"清理垃圾",4,L"清理垃圾");
m_pTabCtrl->InsertItem(4,L"清理痕迹",5,L"清理痕迹");
m_pTabCtrl->InsertItem(5,L"修复IE",6,L"修复IE");
m_pTabCtrl->InsertItem(6,L"高级工具",7,L"高级工具");
m_pTabCtrl->SetSelectedItem(0);
```

4.5 Tab 控件的切换响应

Tab 控件的 item 点击并交换后控件会向窗口抛出自定义消息 DUI_TABMSG_SELCHANGED

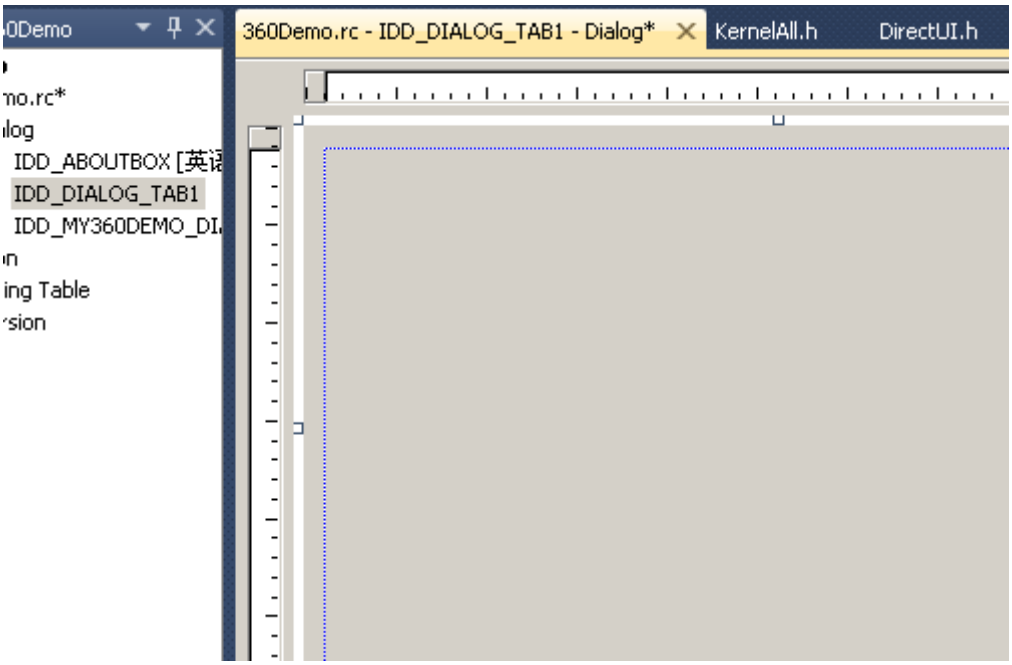
同样的我们处理此消息，消息的 WPARAM 传入的是 item 的 index 序号

```
LRESULT CMy360DemoDlg::OnDUITabSelChanged(WPARAM wparam, LPARAM lparam)
{
    //wParam表示tab项的index
    return 0;
}
```

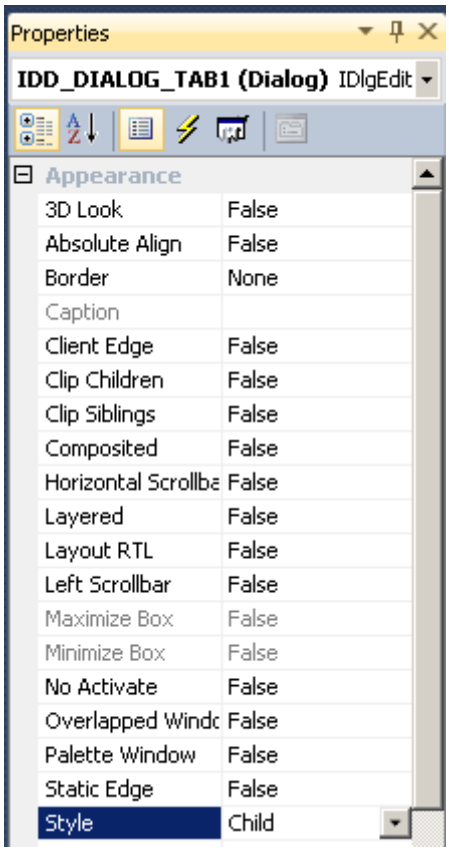
4.6 子窗口的嵌入

DirectUI 控件与其他标准控件可以混合使用，我们通过 HwndObj 控件将其他句柄子窗口 Attach 到此控件上就可以将子窗口放置到 DirectUI 窗口上了，并且这些窗口的布局根据 HwndObj 的布局规则进行变化。下面我们将在主窗口上创建一个子窗口。

1） VS 的资源视图中新建一个对话框，删除其中所有的标准控件



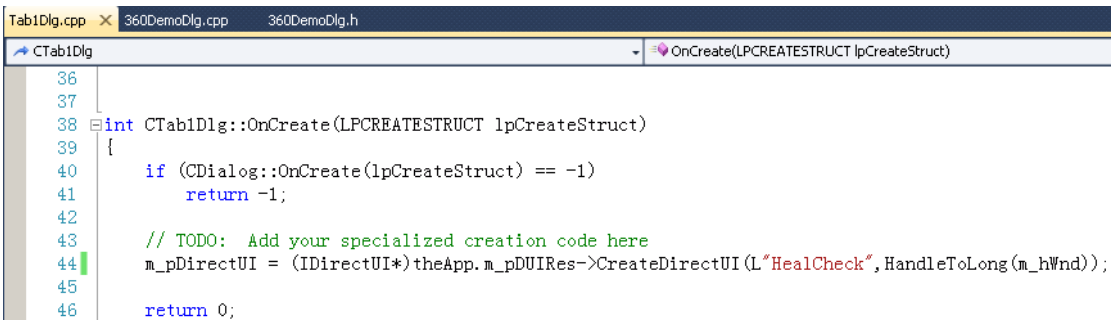
2） 修改此对话框的属性 Border 为 None，Style 为 Child



3） 为此对话框创建一个 CDialog 类 CTab1Dlg

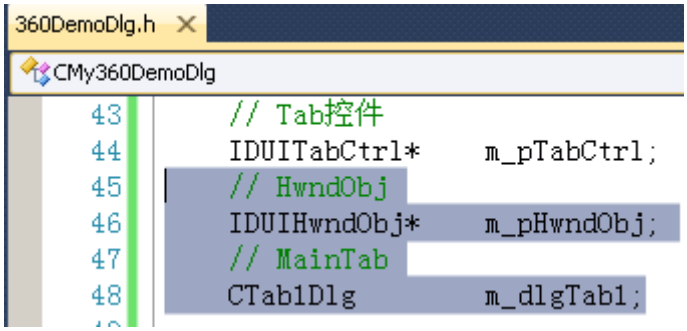
4） 申明一个 IDirectUI 对象指针，并在 OnCreate 中绑定 DirectUI 窗口 HealCheck

m_pDirectUI = (IDirectUI*)theApp.m_pDUIRes->CreateDirectUI(L"HealCheck",HandleToLong(m_hWnd));



5） 在主对话框中申明一个 HwndObj 对象指针和一个子窗口 CTab1Dlg

```
// HwndObj
IDUIHwndObj*    m_pHwndObj;
// MainTab
CTab1Dlg        m_dlgTab1;
```



6） 主窗口 OnCreate 方法中获取 HwndObj 对象指针

```
// HwndObj控件
m_pHwndObj = (IDUITabCtrl*)m_pDirectUI->GetObjectByCaption(DUIOBJTYPE_PLUGIN,
_T("HwndObj1"), TRUE);
```

```
360DemoDlg.cpp  360DemoDlg.h
CMy360DemoDlg                    OnCreate(LPCREATESTRUCT lpCreateStruct)
187     m_pTabCtrl = (IDUITabCtrl*)m_pDirectUI->GetObjectByCaption(DUIOBJTYPE_PLUGIN, _T("TabCtrl"), TRUE);
188     ASSERT(m_pTabCtrl);
189     // HwndObj控件
190     m_pHwndObj = (IDUITabCtrl*)m_pDirectUI->GetObjectByCaption(DUIOBJTYPE_PLUGIN, _T("HwndObj1"), TRUE);
191     ASSERT(m_pHwndObj);
```

在OnCreate中创建CTabDlg窗口并调用HwndObj的attach方法，将这个子窗口Attach到此HwndObj上。

```
// HwndObj控件
m_pHwndObj = (IDUIHwndObj*)m_pDirectUI->GetObjectByCaption(DUIOBJTYPE_PLUGIN,
_T("HwndObj1"), TRUE);
ASSERT(m_pHwndObj);
m_dlgTab1.Create(CTab1Dlg::IDD, this);
m_pHwndObj->Attach((OLE_HANDLE)HandleToLong(m_dlgTab1.GetSafeHwnd()));
m_pHwndObj->ShowWindow((OLE_HANDLE)HandleToLong(m_dlgTab1.GetSafeHwnd()));
```

```
360DemoDlg.cpp  360DemoDlg.h
oDlg                             OnCreate(LPCREATESTRUCT lpCreateStruct)
    m_pTabCtrl = (IDUITabCtrl*)m_pDirectUI->GetObjectByCaption(DUIOBJTYPE_PLUGIN, _T("TabCtrl"), TRUE);
    ASSERT(m_pTabCtrl);
    // HwndObj控件
    m_pHwndObj = (IDUIHwndObj*)m_pDirectUI->GetObjectByCaption(DUIOBJTYPE_PLUGIN, _T("HwndObj1"), TRUE);
    ASSERT(m_pHwndObj);
    m_dlgTab1.Create(CTab1Dlg::IDD, this);
    m_pHwndObj->Attach((OLE_HANDLE)HandleToLong(m_dlgTab1.GetSafeHwnd()));
    m_pHwndObj->ShowWindow((OLE_HANDLE)HandleToLong(m_dlgTab1.GetSafeHwnd()));
```

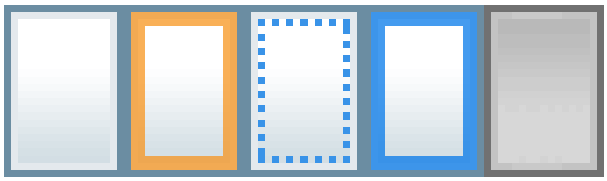
7） 编辑运行，我们可以看到 HealCheck 窗口 Attach 到主窗口中的窗口效果。

四、 DirectUI 界面设计规范

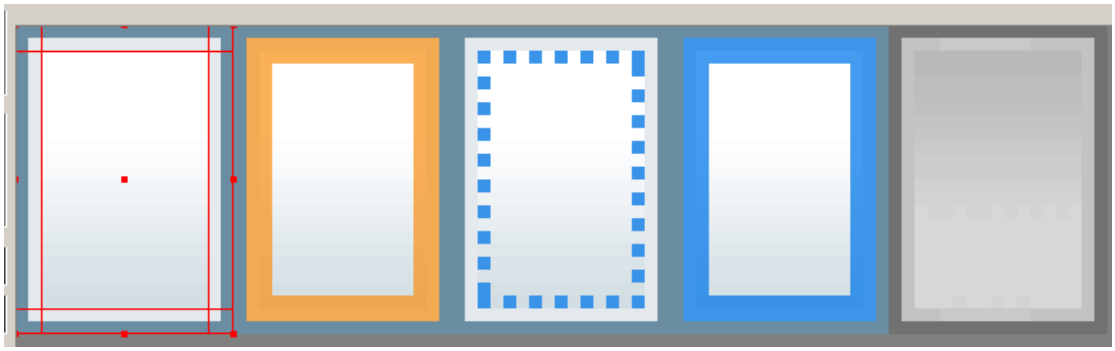
1. 效果图设计规范

2. 控件分割规范

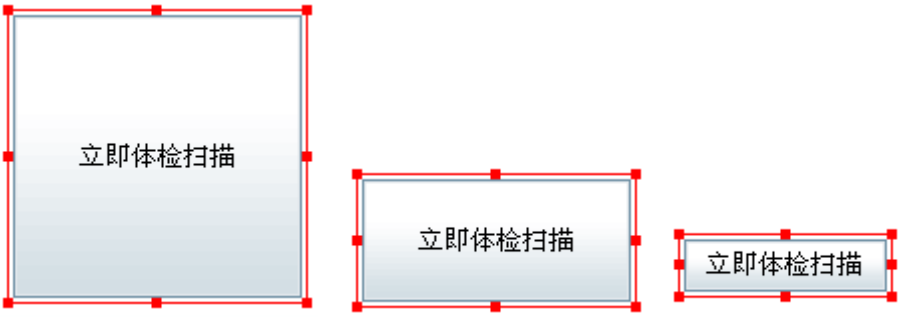
由于 DirectUI 的贴图是支持九宫格拉伸，所以提供的截图需要进行相同像素的合并处理。
如 Button 按钮的切图



一个切图中包含了按钮的五种状态，正常、高亮、按下、激活和禁用。
利用 DirectUI 的九宫的分割绘图我们可以将切图应用到任意大小的控件上。
DirectUI Builder 内 Button 的九宫分割如下



DirectUI Builder 内不同大小的 Button 都可以无失真显示



水平和垂直上相同的像素可以进行合并处理

3. 切图制作规范

不同控件的切图需求列表：

控件名称	状态(Pic 代表一张图片)	切图数量
Button	Pic1:（背景）正常、高亮、按下、禁用、焦点	1

StarCtrl	Pic1:（背景）正常、选中	1
UIForm	Pic1:（背景）正常	1
TabCtrl	Pic1:(Item)、正常、高亮、选中、禁用	1
Combox	Pic1：（背景）正常、高亮、按下、禁用 Pic2：（按钮）正常、高亮、按下、禁用	2
PopupSingleList	Pic1:（背景）正常	1
RadiobBox	Pic1:（未选中）正常、高亮、按下、禁用 （选中）正常、高亮、按下、禁用	1
MenuBar	Pic1：正常、高亮、按下、禁用 Pic2：背景	2
LogoObj	Pic1:背景	1
CheckBox	Pic1:（未选中）正常、高亮、按下、禁用 （选中）正常、高亮、按下、禁用 （半选）正常、高亮、按下、禁用	1
ScrollBar	Pic1：（按钮）（垂直）（上方）正常、高亮、按下、禁用(下面) 正常、高亮、按下、禁用（水平）（左边）正常、高亮、按下、禁用（右边）正常、高亮、按下、禁用 Pic2：（滑槽）(垂直)正常 Pic3：（滑槽）(水平)正常 Pic4：（滑块）（垂直）正常、高亮、按下、禁用 Pic5：（滑块）（水平）正常、高亮、按下、禁用 Pic6：（Tick）（垂直）正常 Pic7：（Tick）（水平）正常	7
HeaderCtrl	Pic1：（Item）正常、高亮、按下 Pic2：（背景）正常	2
ProgressBar	Pic1:背景、选中	1
SliderBar	Pic1：（滑槽）（水平）正常 Pic2：（滑槽）（垂直）正常 Pic3：（滑块）（水平）正常、高亮、按下、禁用 Pic4：（滑块）（垂直）正常、高亮、按下、禁用 Pic5：（滑块）（指向上方）正常、高亮、按下、禁用 Pic6：（滑块）（指向下方）正常、高亮、按下、禁用 Pic7：（滑块）（指向左方）正常、高亮、按下、禁用 Pic8：（滑块）（指向右方）正常、高亮、按下、禁用	8
OutLookBar	Pic1：（背景）正常 Pic2：（抽屉）正常、高亮、选中、禁用 Pic3：（Item）正常、高亮、选中、禁用	3

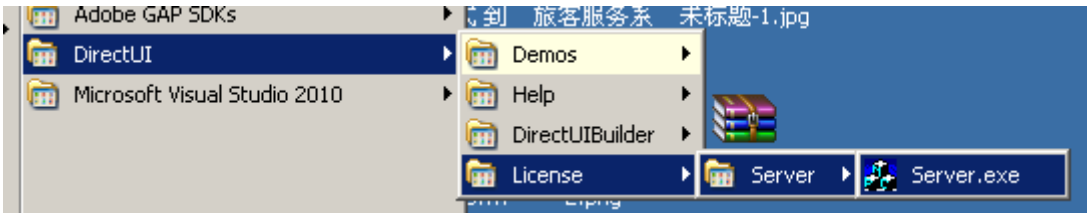
SimpleTree	Pic1：（背景）正常 Pic2：（Item）正常、高亮、选中、禁用 Pic3：（展开按钮）展开、收缩	3
TaskPanel	Pic1:（背景）正常 Pic2:（抽屉）正常、高亮、选中、禁用 Pic3:（Item）正常、高亮、选中、禁用	3
PopupMenu	Pic1：（背景）正常 Pic2：（Item）正常、高亮、选中、禁用 Pic3：（Radio）选中 Pic4：（CheckBox）选中、未选中 Pic5：（分隔条）正常	5
FormBorder	Pic1:（标题栏）正常、非激活 Pic2:（左边框）正常、非激活 Pic3:（右边框）正常、非激活 Pic4:（下边框）正常、非激活 Pic5:（关闭按钮）正常、高亮、按下、禁用、焦点 Pic6:（最大化按钮）正常、高亮、按下、禁用 Pic7:（还原按钮）正常、高亮、按下、禁用 Pic8:（最下化按钮）正常、高亮、按下、禁用、焦点 Pic9:（帮助按钮）正常、高亮、按下、禁用、焦点	9
ToolBar	Pic1：（Item）正常、高亮、按下、禁用、焦点 Pic2：（扩展按钮）正常、高亮、按下、禁用、焦点 Pic3：（背景）正常 Pic4：（分隔条）正常	4

五、 DirectUI Builder 皮肤制作介绍

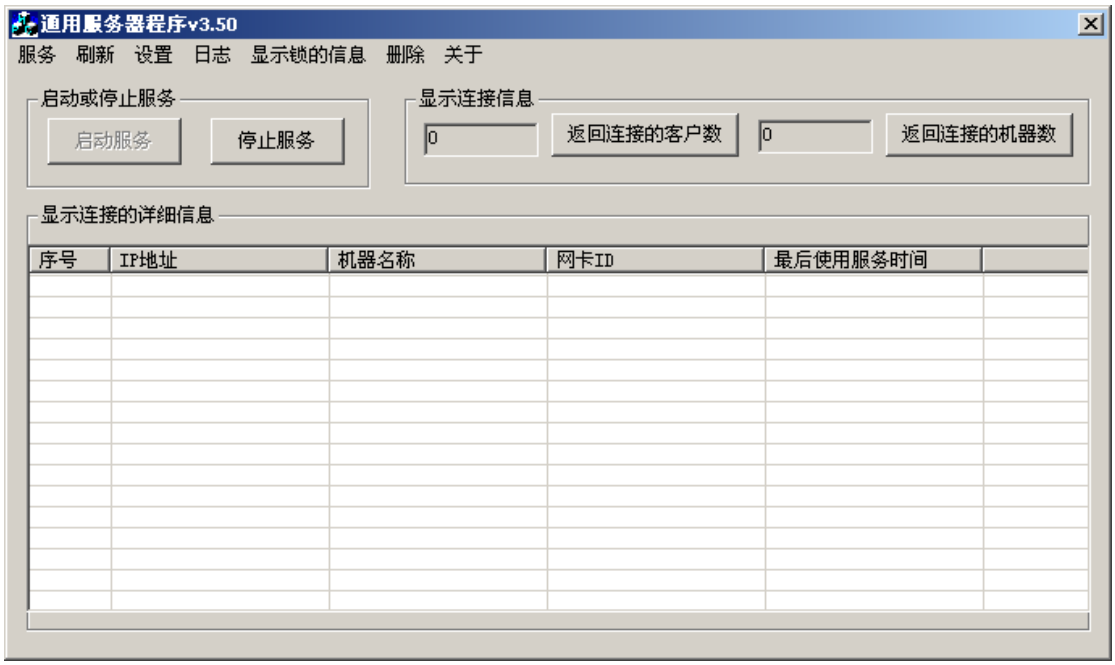
1. 启动 DirectUI Builder

1）由于 DirectUI Builder 采用了硬件狗的加密方式，所以启动前先插入硬件狗；

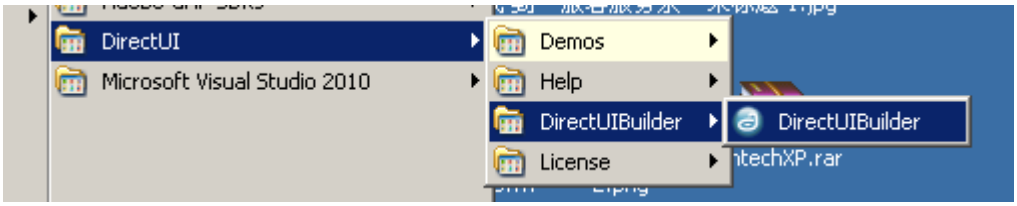
2）然后需要先打开硬件狗许可程序；



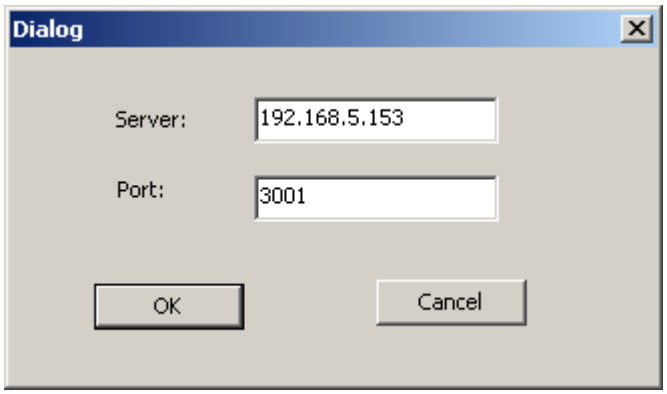
3）启动会出现许可管理程序后会自动最小化到托盘，在 DirectUIBuilder 运行构成中请不要关闭许可管理程序。



DirectUI Builder 是 DirectUI 界面库的皮肤制作工具，安装好 DirectUI 开发包后会在桌面与程序组中出现 DirectUI DirectUI 的快捷启动程序。



第一次启动 DirectUIBuilder 会出现许可管理器配置窗口，需要分别填写许可管理器在局域网内的地址，然后填写访问端口为 3001。



点击“OK“按钮，如果 DirectBuilder 正常访问到许可管理器后将会打开 DirectUIBuilder。

2. DirectUI Builder 介绍

2.1 主界面

2.2 菜单区

文件菜单：

1、 新建、打开、保存与关闭解决方案 dsln；

2、 新建与导入 DUI 文件；

3、 新建与导入 Skn 文件；

编辑菜单：

1、 可以对选中的控件对象进行拷贝、粘贴、剪切、删除操作；

2、 可以对当前的操作进行连续的撤销与重做。

视图菜单：

可以显示与隐藏各种工具栏、状态栏、解决方案、工程区、资源区、属性区、工具箱区与样式区。

工具菜单：

可以启动多语种管理、设置皮肤密钥、输出皮肤、主题管理器

帮助菜单：

弹出关于 DirectUIBuilder 的对话窗口。

主工具栏：

包含新建、打开与保存解决方案、撤销与重做、剪切、拷贝与粘贴、ZOrder 坐标调整、DirectUI 对象创建、DirectUI 对象最小化设置、还原设置、Help 属性设置、关于 DirectUI。

对齐工具栏：

可以对选中的一个或多个对象进行各种对齐属性的设置。

1、 所有选中靠左对齐、靠右对齐、靠上对齐、靠下对齐；

2、 所有选中控件的纵向与横向的居中布局；

3、 所有选中控件的横向与纵向的平均分布；

4、 所有选中控件的宽度、高度、大小完全保持一致。

布局工具栏：

可以对选中的对象进行快捷的布局。从左到右的图标的含义分别是： 左上、右上、左下、右下、左边、右边、顶边、底边、充满、居中。

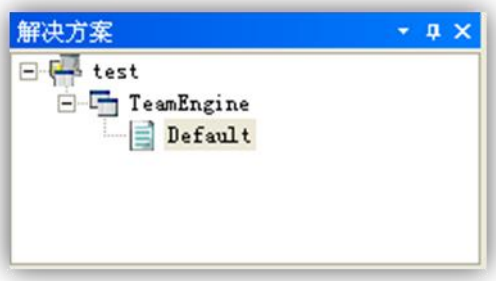
语种选择工具栏：

可以根据当前界面解决方案中设置的语种个数，列举出所有的语种供选择，选中一个语种后当前就使用该语种进行文字的设置。

主题选择工具栏：

列举出当前 DirectUIBuilder 环境设置中设置的所有主题的名称。选中一个主题后，当前的 SKN 就使用该主题。当界面解决方案有多个 SKN 同时打开时，SKN 之间的切换会导致该主题选择列表的当前主题的动态变换。即一个 SKN 文件对应一个主题。多个 SKN 文件可以使用相同主题。

2.3 解决方案区



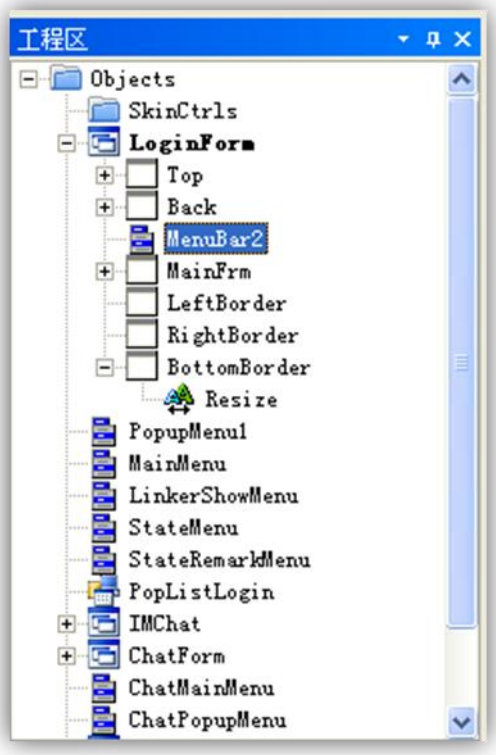
- 该区显示解决方案.dsln 的树状结构，包括.dui、.skn、.dtheme.

```
--dsln
--dui(模块)—(相当于 visualstudio 中的.dsp)
    --skn1
    --skn2
--dui(模块)—(相当于 visualstudio 中的.dsp)
    --skn1
    --skn2
```

树状结构如下：

- 用户可以右键点击 dui 节点弹出菜单，可以增加、移除 dui。
- 用户可以右键点击 skn 节点弹出菜单，可以增加、移除 skn。
- 用户可以点击该区标题的图钉按钮进行该区的固定与停靠。也可以点击关闭按钮隐藏该区，隐藏后可以通过工具菜单再次显示该区。

2.4 工程区



- 工程区显示的是 dui 文件的内容。描述了一个程序中所有的窗口(DirectUI 对象)和窗口中所包含的控件名称（Controls）。DirectUI 对象与控件、控件与控件之间都是树状结构，即他们之间属于父与子的关系，层层迭代形成一个复杂的界面控件组合。在工程 dui 文件第一次打开时，DirectUI 对象左边没有“+”号，代表其在刚加载进来后并没有一次性将所有的该对象里面所有的对象都读入进来。这样的好处是可以大大提高打开 dui 文件时的加载速度与内存占用。

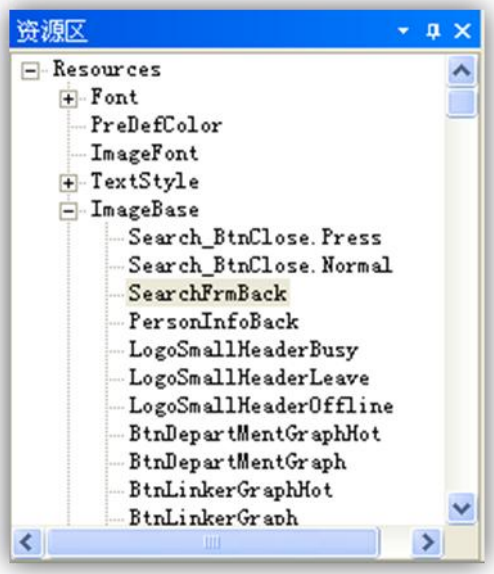
  树状结构如下：

```
-mainframe (DirectUI 对象)
    -background (UIForm 对象)
        -button1
        -button2
    -bottom (UIForm 对象)
        -static1
        -static2
-aboutform (DirectUI 对象)
    -back (UIForm 对象)
        -top (UIForm 对象)
        -middle (UIForm 对象)
```

- 用户可以单击 DirectUI 对象来预览其界面。此时 DirectUIBuilder 将读取该对象下所有子控件，递归读取直到最后。所以用户点击后 DirectUI 对象左边会变成“+”号。代表其子对象已经全部读取进来。当用户在工作区直接点击某个对象后，控件树将自动展开并定位到当前选中对象的节点。

- 用户可以点击该区标题的图钉按钮进行该区的固定与停靠。也可以点击关闭按钮隐藏该区，隐藏后可以通过工具菜单再次显示该区。

2.5 资源区

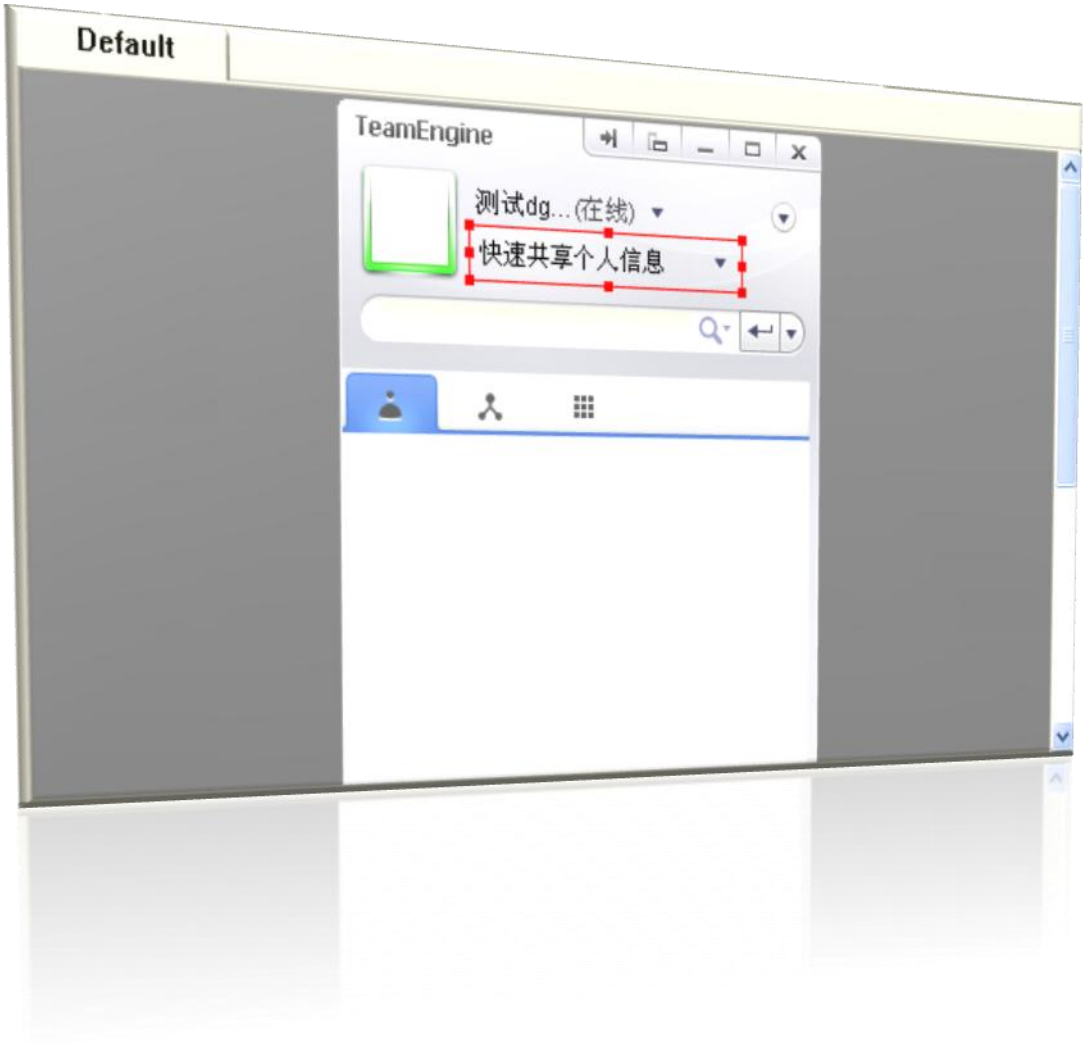


这个区将当前 SKN 文件中用到的所有的共享资源（Image、ImageSection、TextStyle、Font 等）以分组的方式进行列举。

```
-Font
    -font1
    -font2
-Image
    -image1
    -image2
```

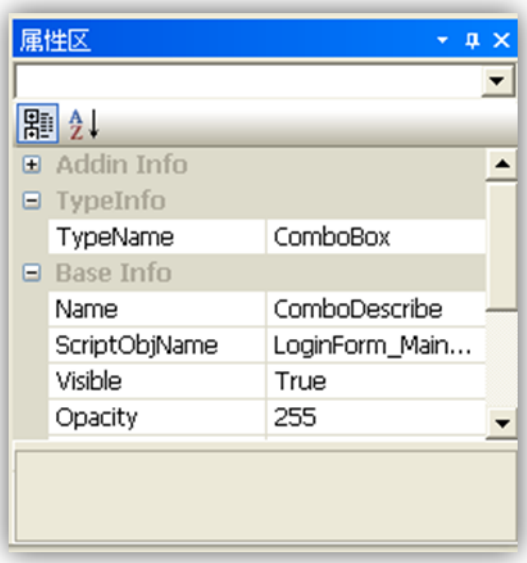
分组结构如下：

2.6 视图工作区



- 该区域是 DirectUIBuilder 最重要的部分，集界面预览、控件拖拽、复制粘贴、控件停靠、控件对齐等功能一身的工作区。该部分功能是否易用与用户对其掌握程度直接影响到界面制作的效率。
- 控件的外围有红框代表该控件在选中状态，用户可以通过拉伸红框的四个边来对控件进行大小的调整，也可以移动红框来对控件的位置进行调整。用户既可以单选一个控件进行操作，也可以通过按住 Shift 键来多选几个控件进行操作。当控件处在选中状态时，可以按方向键对控件的位置进行微调。也可以通过按住 Shift 键来对单个选中的控件进行大小的微调。
- 该区域的上方有一个 Tab 控件，一个 Tab 页面对应一个 SKN。用户可以通过切换 Tab 页对 SKN 进行切换。

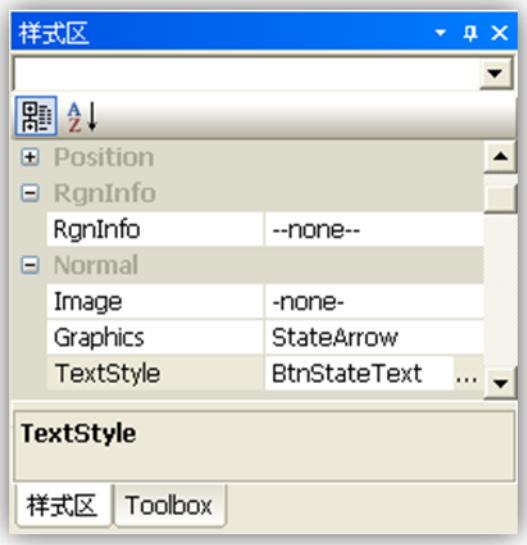
2.7 属性区



- 每个控件的属性均在属性区进行展示。
- 常用的操作有属性分类节点的展开与收起。属性分类节点的展开与否的状态被记录在 DUI 文件中。所以用户打开 DUI 文件后可以重现上次操作的状态。对于某些控件具有复杂属性分类并属性比较多的情况下，该功能可以提供用户编辑属性的效率。
- 属性区的值与 DUI 文件对应。
- 用户可以点击该区标题的图钉按钮进行该区的固定与停靠。也可以点击关闭按钮隐藏该区，隐藏后可以通过工具菜单再次显示该区。

2.8 样式区

截屏如下：



- 每个控件的图像选区、文本风格、颜色等设置都在样式区进行操作。
- 其分类节点的展开与否的功能与属性区一样，其状态保存在 SKN 文件中。
- 样式区的属性与 SKN 文件对应。
- 用户可以点击该区标题的图钉按钮进行该区的固定与停靠。也可以点击关闭按钮隐藏该区，隐藏后可以通过工具菜单再次显示该区。

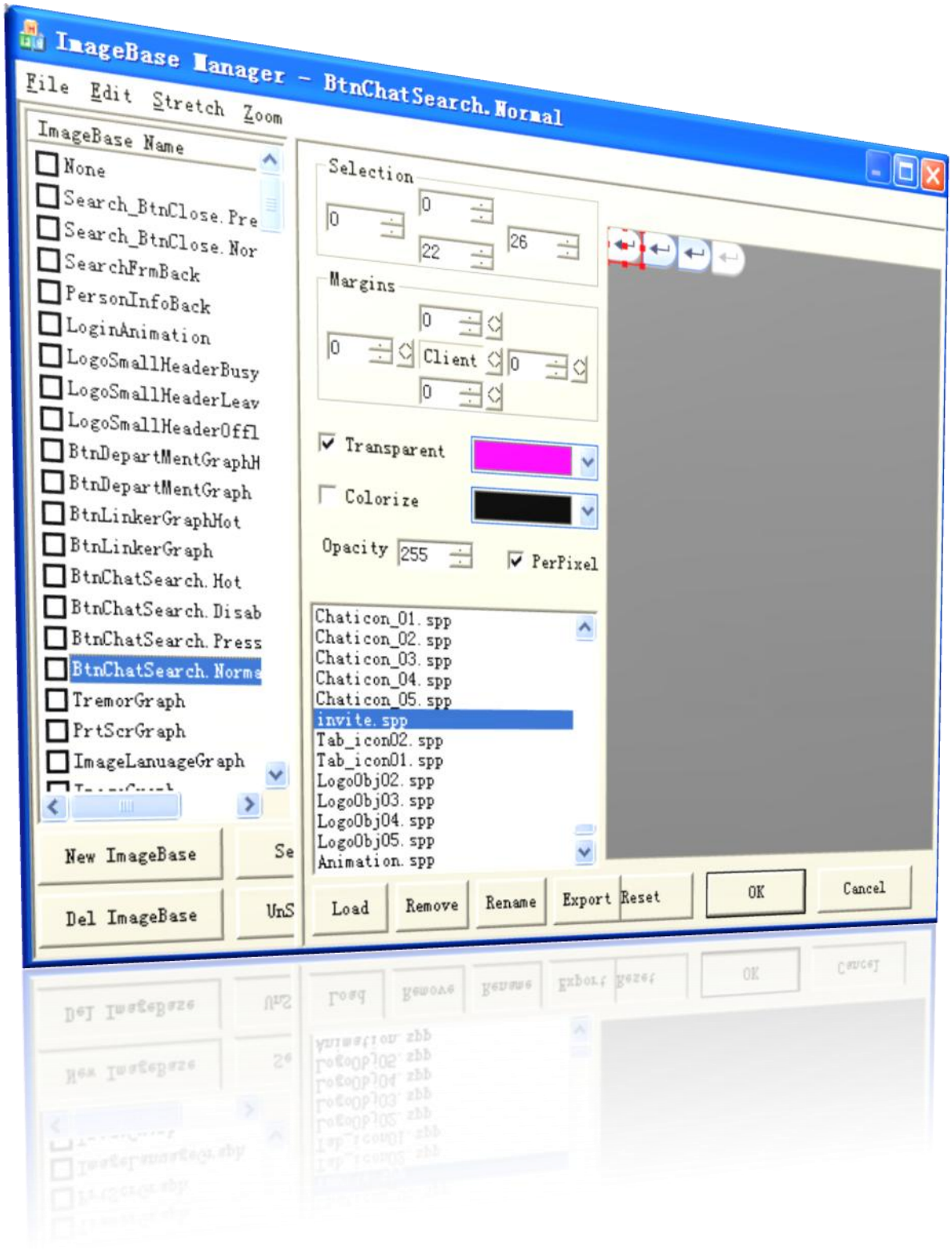
2.9 控件工具箱区

截屏如下：

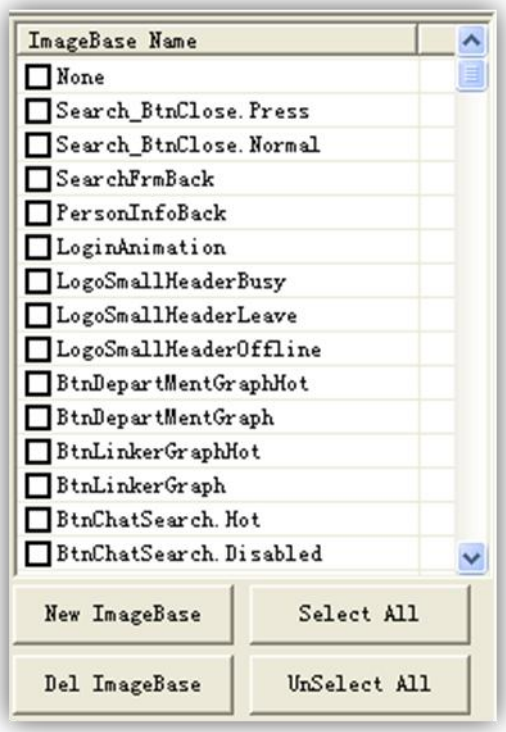
- DirectUI 控件采用插件方式开发与管理。用户可以编写自己的 DirectUI 控件。也可以建立自己的控件分类。工具箱中的每一个控件均有控件的图标、控件的名称、控件的分类、控件拖放时的光标。用户在开发控件需要将这些信息告诉 DirectUIBuilder。
- 目前 DirectUIBuilder 中的自带控件分以下四类：Kernel 控件组、Office 控件组、Advanced 控件组、工业控件组。
- 用户可以点击该区标题的图钉按钮进行该区的固定与停靠。也可以点击关闭按钮隐藏该区，隐藏后可以通过工具菜单再次显示该区。

2.10 图像选区界面

该界面分 2 大部分：ImageBase 管理区与图像选区编辑区。



1） ImageBase 管理区

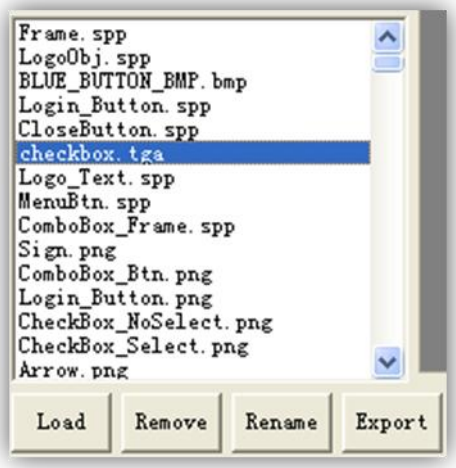


- ImageBase 区管理当前 SKN 文件中所有的 ImageBase，用户可以对其进行创建、编辑与删除等操作。
- ImageBase 的命名不能重复。
- ImageBase 的命名使用英文或数字的组合，不能使用特殊字符。
- ImageBase 的命名尽量做到容易读懂。
- 通过列表前面的复选框，用户部分选择或全选，进行删除操作。

2） 图像选区编辑区

ImageBase 对象包含选择哪张图片，选中该图片的哪部分区域，选中区域图片的四边的拉伸与平铺属性，该区域图片的透明色等信息，所以在该区域包含以下部分：

3） 图像列表：



- 在该列表中列举了当前 SKN 文件中所有的图片文件。
- 如果要设置的图片不再当前的 SKN 文件中，用户可以从外部选择一张图片进来。
- 用户可以选中一张图片后进行移除，这样 SKN 文件中不再包含该文件。
- 当选中其中的一张图片后，右边预览区即显示该图片。

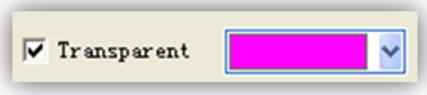
4） 区域选择区：

- 该区域有四个值，分别对应选区的左边、顶边、右边、底边。
- 左边值不小于 0，顶边值不小于 0，右边值不大于图片的宽度，底边值不大于图片的高度。
- 该区域指定 ImageBase 选择当前图片的某个区域作为其图像源。
- 用户可以在图片预览区直接操作图片选框来大致选择图像选区。
- 用户也可以通过点击四个值旁边的 Spin 控件，对选区进行微调。

5） 图片不被拉伸区：

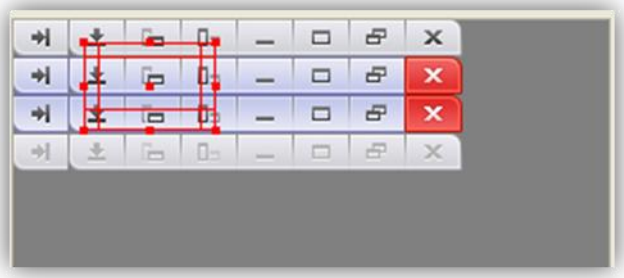
- 该区域有四个值，分别指定了图像中不能被拉伸的 4 条边的大小。分别对应左边、顶部、右边、底部。
- 用户可以通过点击四个值旁边的 Spin 控件，对不被拉伸区域进行微调。
- 在每个方向的边上与中间区域还有拉伸与平铺的互斥属性，他们用于指定 5 个缩放区域是否允许拉伸。
- 用户可以点击四个值旁边的 CheckBox 控件来设置拉伸或平铺。CheckBox 被选中表示该边属于拉伸状态，否则属于平铺状态。
- 用户也可以通过选中拉伸菜单的 Tile All 菜单项来对 5 个部分全部设置为平铺，选中 Stretch All 菜单项来对 5 个部分全部设置为拉伸。

6） 透明色设置区：



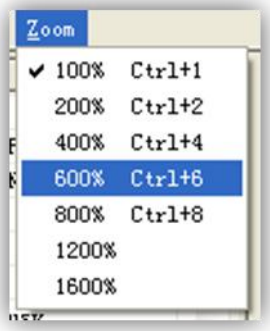
- 是否选中透明色前面的 CheckBox，表示在该图片使用是否采用透明色。
- 如果该图片中不存在透明色的概念，请不要勾选该 CheckBox。这样会提高 DirectUI 绘图的效率。
- 如果勾选了 CheckBox，则需要设置透明色的 RGB 值。

7） 图像预览区：



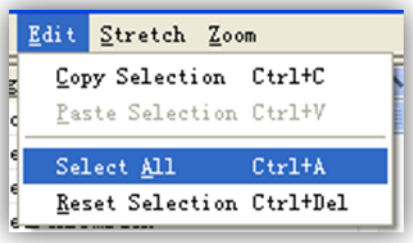
- 该区可以对选中的图片进行显示。
- 可以对显示的图像进行放大与缩小。
- 可以通过鼠标直接拖拽图像选框来确定选区位置与大小。
- 对于选框的拖拽的范围作了边界处理，避免用户的误操作。

8） 缩放菜单：



- 可以对图像进行 100% ~ 1600%的缩放
- 图像缩放后，图像选框也跟着一起缩放，同时选框的位置与大小的调整保持以像素为单位。

9） 拷贝与粘贴选区菜单：



- 拷贝一个 ImageBase 中的选区信息（包括选中的图像、选区位置与大小、不被拉伸区域的数值与拉伸与否的状态、是否透明与透明色）。

- 在另外一个 ImageBase 中执行粘贴选区，则将选区信息拷贝了过来。
- 一般在一个控件具有多个状态时需要快速的设置其 ImageBase 时采用。由于通常情况下，我们将相同控件的图像放在一个图片文件中，同时他们的选区相对靠近，其他信息又几乎一致，此时使用拷贝粘贴功能可以尽量避免重复操作，大幅度提供设置 ImageBase 的速度。

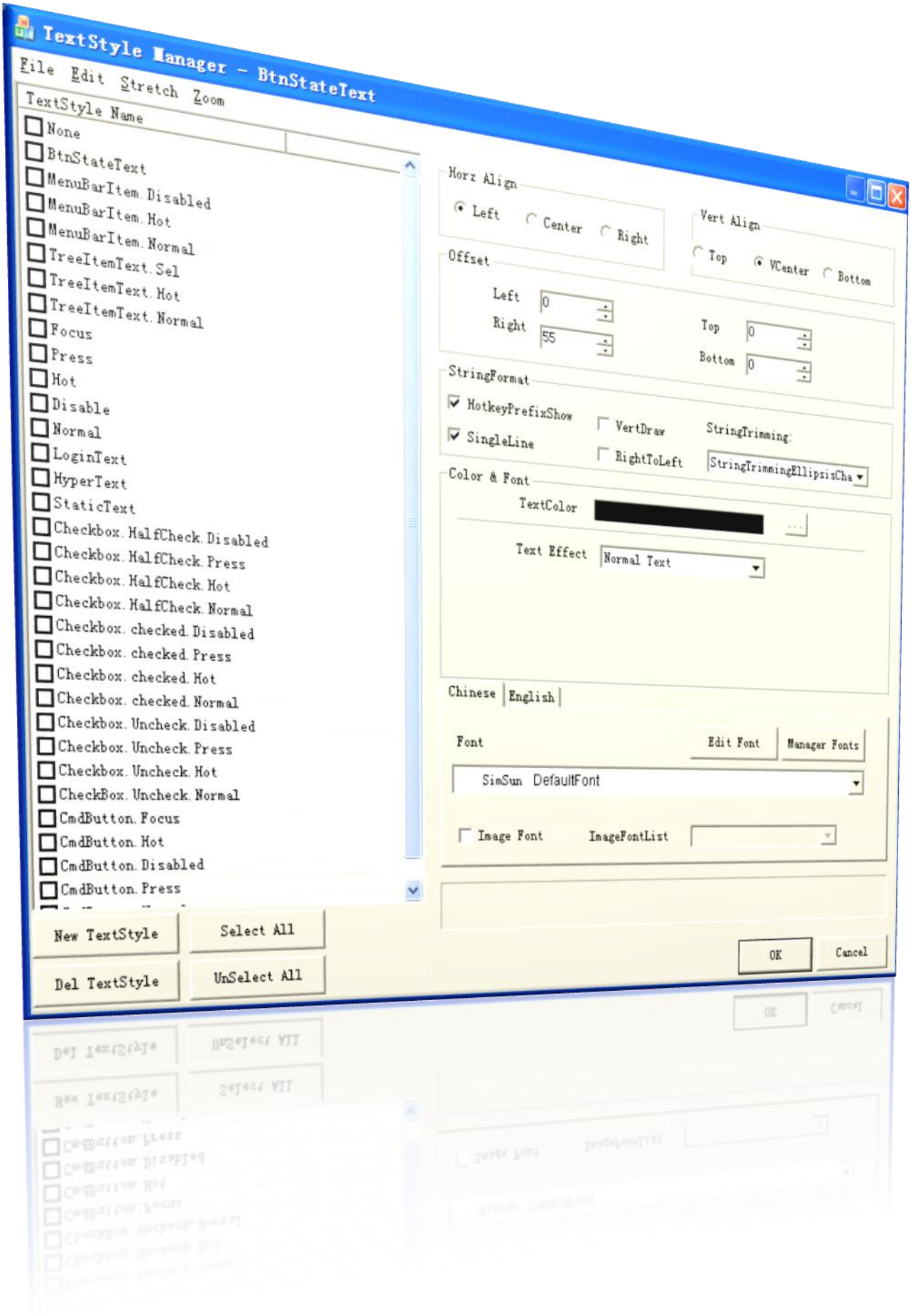
10）控制按钮区：

- 在该区包含 Reset、OK、Cancel3 个按钮
- Reset 用于初始化当前 ImageBase 的选区信息：
  - 图片选择为无
  - 选区值：left = 0, top = 0, right = 20, bottom = 20.
  - 不被拉伸区域：left = 0, top = 0, right = 0, bottom = 0

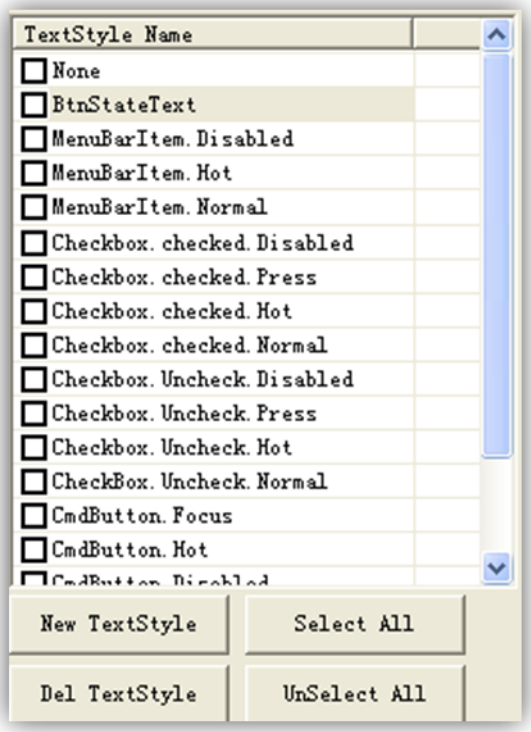
    不被拉伸区域所有边为平铺状态
  - 是否使用透明色 CheckBox 为不勾选，透明色值：Red = 255,Green = 0,Blue = 255
- OK 按钮为保证当前对 ImageBase 所做的修改，并对样式区的属性值设置为当前选中的 ImageBase。
- Cancel 按钮为保证当前对 ImageBase 所做的修改，但不会对样式区的属性值设置进行改变。
- 若要将样式区属性值设置为空，要先选中 ImageBase 列表中第一个 NONE 项，然后点击 OK 按钮。

2.11 文本风格设置界面

该界面分 2 大部分：TextStyle 管理区与文本风格设置区。



1） TextStyle 管理区



- TextStyle 区管理当前 SKN 文件中所有的 TextStyle，用户可以对其进行创建、编辑与删除等操作。
- TextStyle 的命名不能重复。
- TextStyle 的命名使用英文或数字的组合，不能使用特殊字符。
- TextStyle 的命名尽量做到容易读懂。
- 通过列表前面的复选框，可以部分选择或全选，进行删除操作。

2） 文本风格设置区

TextStyle 对包含文本的横向与纵向的对齐方式、横向与纵向的偏移量、字符串格式、文字效果、文字前景颜色、文字阴影颜色、多语种字体设置等信息，所以在该区域包含以下部分：

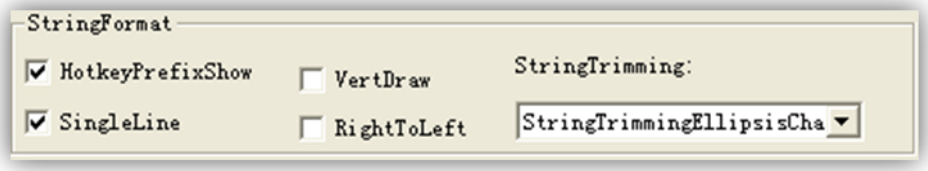
3） 对齐方式：

- 横向：靠左、横向居中、靠右
- 纵向：靠上、纵向居中、靠下

4） 偏移量：

- 横向偏移：left,right 数值可以对上面的横向对齐方式进行微调
- 纵向偏移：top,bottom 数值可以对上面的纵向对齐方式进行微调

5） 字符串格式：



字符串格式对字符串的转义符的绘制、单行与多行、垂直绘制、从右到左绘制、字符串截断等进行了设置

- 转义符：HotkeyPrefixShow 复选框选中代表字符串中的“&”符号转义成下划线，如：&File ,DirectUI 将绘制成：File

  复选框未选中代表字符串中的“&”符号不进行转义，上面的那个例子将绘制成：&File
- 单行与多行：SingleLine 复选框选中代表字符串以单行的形式绘制，否则以多行的形式绘制。
- 垂直绘制：VertDraw 复选框选中代表以垂直方向绘制字符串。一般在纵向 TabCtrl 的 Tab 标签中的文字使用该选项。
- 从右到左绘制：RightToLeft 复选框选中代表以从右向左的方向绘制文本。一般在阿拉伯系统中用到。
- 超长后截断：当文字的长度超过控件所能显示的范围则需要对字符串进行截断，而对控件进行截断则有如下的模式：

  StringTrimmingNone：不做截断，超出范围的部分由用户来处理。有可能用户看到半个字符结尾的情况。

  StringTrimmingCharacter:以字符为单位的截断，但在截断后的文本中没有省略号“…”。

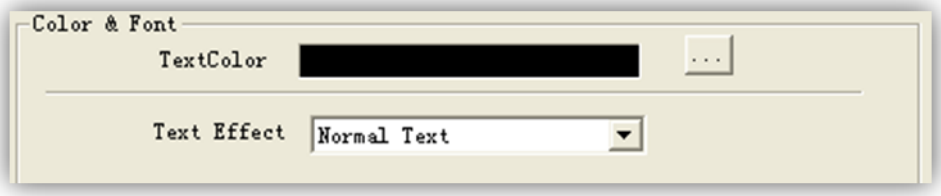
  StringTrimmingWord:以单词为单位的截断，，但在截断后的文本中没有省略号“…”。

  StringTrimmingEllipsisCharacter: 以字符为单位的截断，并在截断后的文本中带有省略号“…”。

  StringTrimmingEllipsisWord: 以单词为单位的截断，并在截断后的文本中带有省略号“…”。

  StringTrimmingEllipsisPath:以路径为单位的截断，并在截断后的文本中带有省略号“…”。

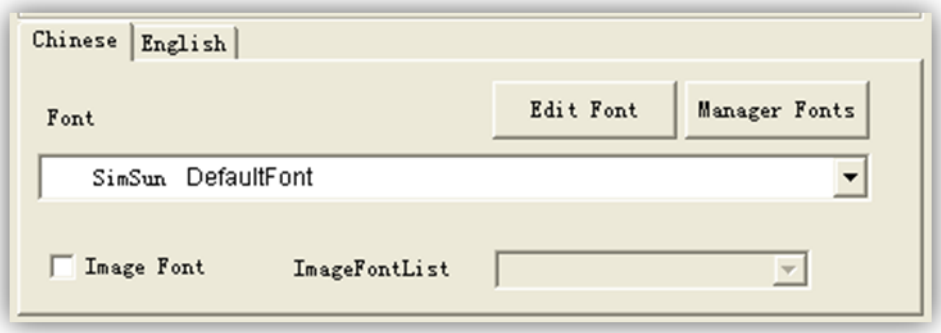
6） 文字效果：



DirectUI 支持以下几种的文字效果：

- Normal：正常文本绘制，不带任何特殊效果。
- Shadowed：投影文本绘制，分前景层与投影层，用户可以分别设置 2 者的颜色 RGB 值。
- Outlined：轮廓式文本绘制，分正常文字层与轮廓层，用户可以分别设置 2 者的颜色 RGB 值。
- Shadowed On PerPixel：类似与 Vista 与 Windows7 标题栏文本的绘制方式，用户可以设置 2 者的颜色，还可以设置模糊晕层的增量值，默认为 3。

7） 多语种字体：

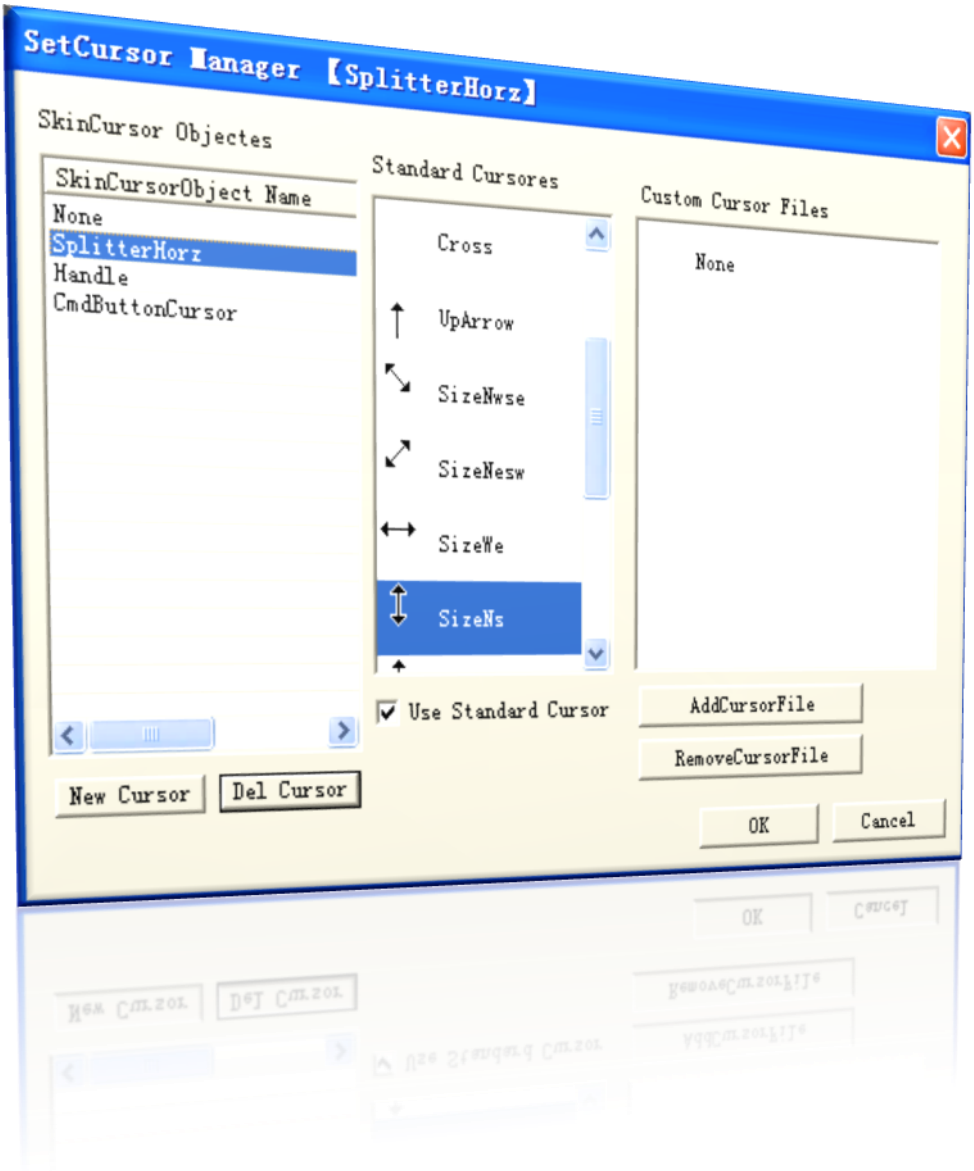


- 由于字体与界面当前使用的语种有关系，所以 DirectUI 的文本字体信息是从属于当前语种的。
- 当前界面方案中有多少个语种，在该区就会出现多少个语种 Tab 页面，在每个 Tab 页面可以设置当前文本风格所用字体。
- 用户同时还可以创建、编辑、删除字体。
- 在某些应用场合，还需要用到图片式字体，例如 LED 数字，或特定样式的英文字符，可以创建 ImageFont。当选中 Image Font 复选框后，右边的 ImageFontList 则使能，用户可以在这里选择一个字体来使用。此时，上方的标准型字体则为禁用状态。

8） 效果预览区：

该区对上面的设置进行即时的更新，使得各项配置的改变，都能及时看到最新配置的效果。

2.12 光标资源设置界面

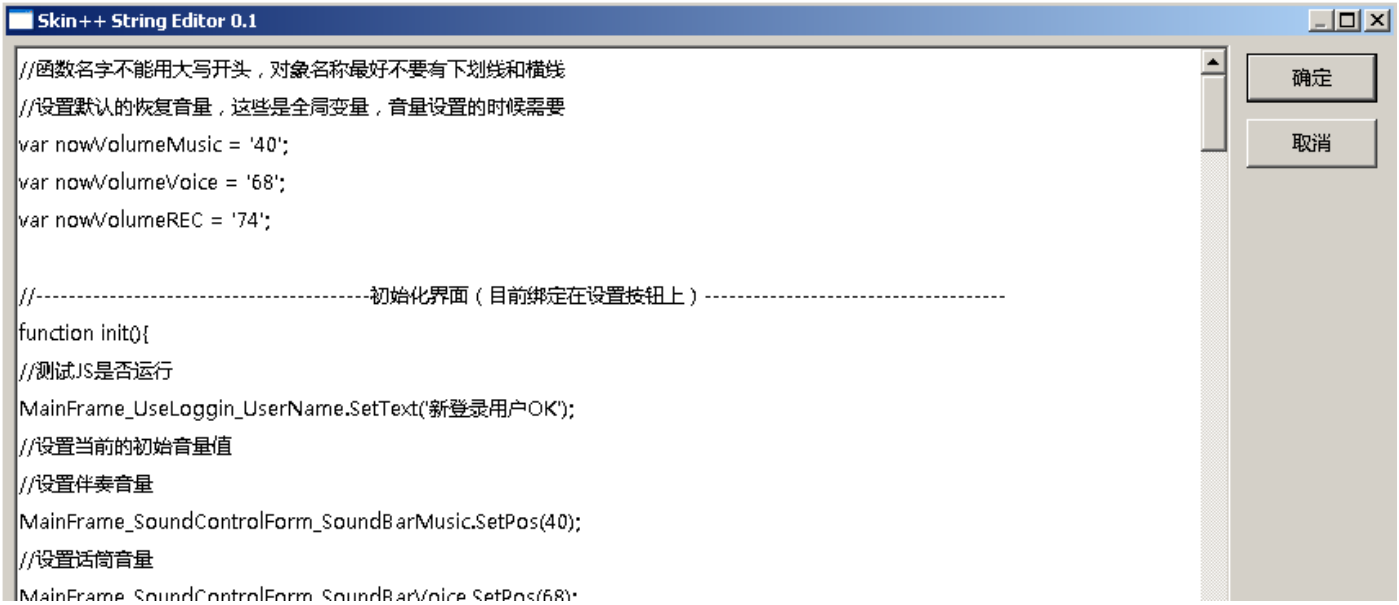


该界面包含 SkinCursor 管理区、SkinCursor 设置区。

- SkinCursor 管理区：可以新建、删除、编辑某项 SkinCursor 对象。
- SkinCursor 设置区：设置区分系统标准光标与自定义光标文件。勾选 Use Standard Cursor，则使用系统标准光标，否则使用自定义光标。
  - 系统标准光标：Windows 系统标准光标在中间的列表中进行了枚举，可以选择其中一项。
  - 自定义光标文件：点击 AddCursorFile 按钮来添加一个自定义光标文件（支持.cur,.ani 2 种格式），点击 RemoveCursorFile 按钮来移除一个自定义光标文件。

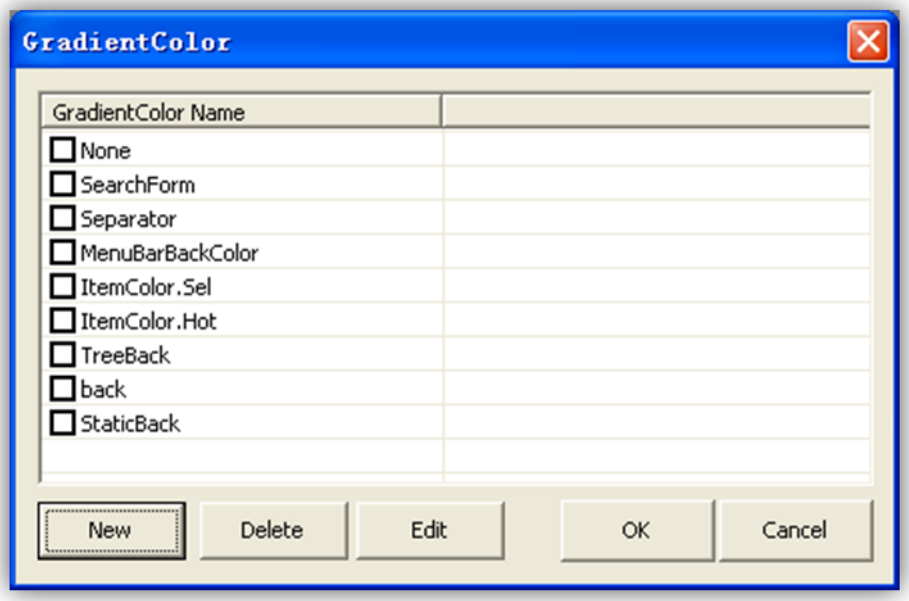
2.13 脚本编辑界面

DirectUI 支持 JavaScript 脚本调用，可以在该界面写入控件的脚本代码，实现界面逻辑与业务逻辑的分离。



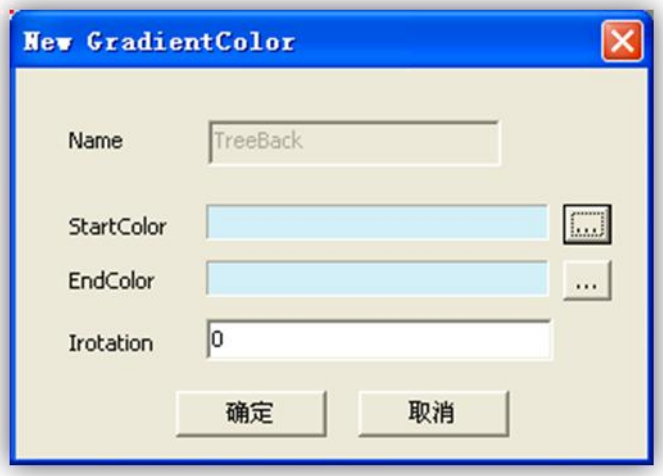
在 DirectUI Builder 中选择控件，并切换到属性区，点击 ScriptCode 即可打开脚本编辑面板。

2.14 颜色管理界面



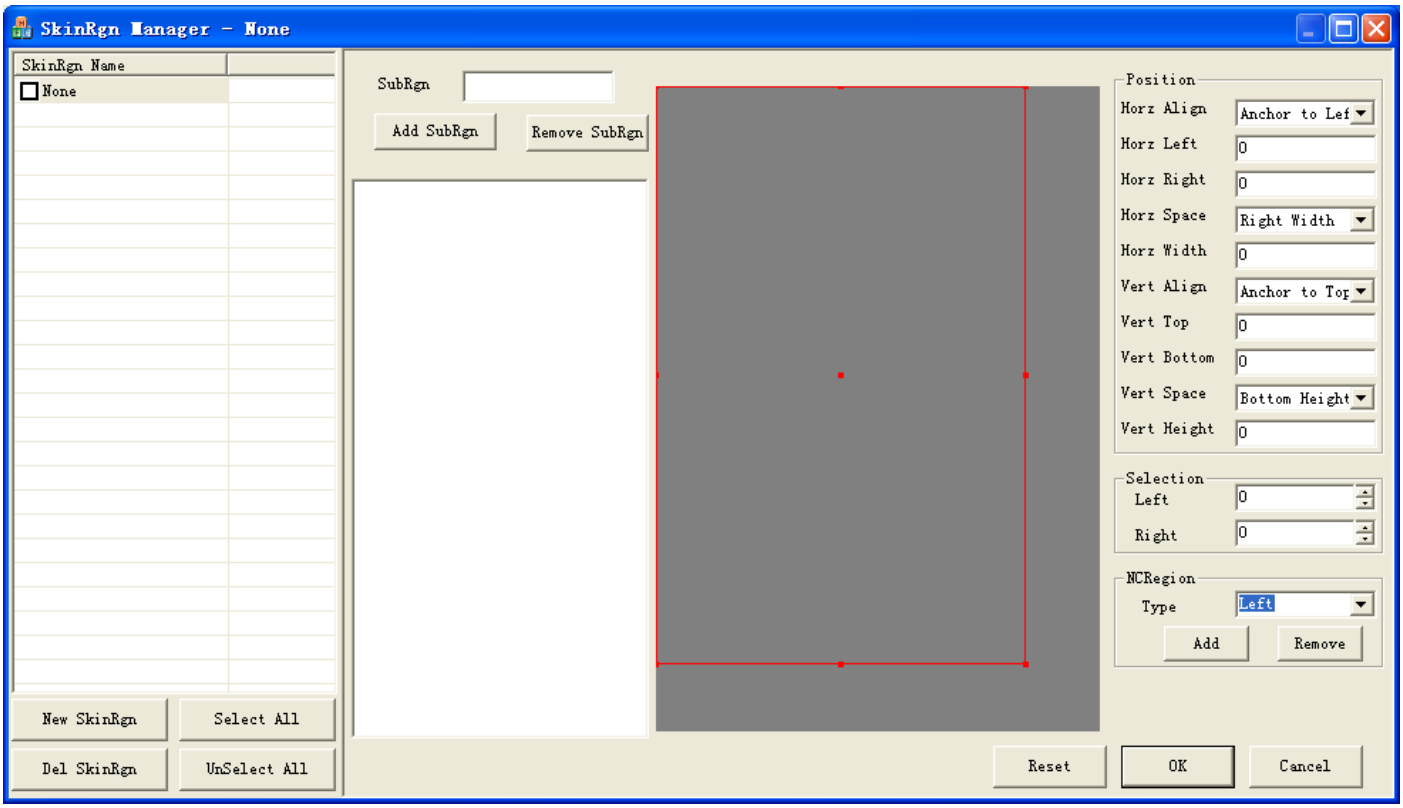
点击某个控件样式属性中的 Color，则弹出上面这个对话框。上部是 GradientColor 的列表，下部有新建、删除和编辑 GradientColor 的命令按钮。用户选中其中一项后点击 OK 按钮，则样式属性 Color 的值更新为选中那一项。如果选中 None 一项，点击 OK 按钮后，样式属性 Color 的值则显示为--none--。

当用户选中某项后，点击 Edit 按钮，则弹出下面窗口：



- Name：显示被编辑 GradientColor 的资源名称。
- StartColor：渐变色的开始颜色。
- EndColor:渐变色的结束颜色。
- Irotation:渐变方向的角度。0° 为从上到下渐变，180° 为从左到右渐变。

2.15 区域管理界面



该界面分分 2 部分：SkinRgn 管理区和 SkinRgn 编辑区

1） SkinRgn 管理区

- SkinRgn 区管理当前 SKN 文件中所有的 SkinRgn 对象，用户可以创建、编辑与删除等操作。

- SkinRgn 的命名不能重复。
- SkinRgn 的命名使用英文或数字的组合，不能使用特殊字符。
- SkinRgn 的命名尽量做到容易读懂。
- 通过列表前面的复选框，可以部分选择或全选，进行删除操作。

2） SkinRgn 编辑区

该区分 3 个部分：子区域列表、中间预览区、子区域的属性

3） 子区域列表

- 在一个 SkinRgn 对象中包含若干个子区域 SubRgn。
- 在 SubRgn 编辑框中填写新创建子区域的名称，然后点击 Add SubRgn 按钮，即添加一个子区域。
- 在子区域列表中选中一项，点击 Remove SubRgn 按钮，即删除一个子区域。

4） 中间预览区

- 预览区显示当前控件的图像
- 控件图像上显示子区域及不被拉伸区，显示随子区域列表选中项的改变而改变。
- 在该预览区不支持鼠标拖拽选区的功能。要调整其选区位置，需要调整子区域属性区的相应的数值。

5） 子区域属性区

该区域包含 3 部分：子区域布局、不被拉伸区域、非客户区拉伸类型

6） 子区域布局

该区域的布局概念同控件的布局。布局父对象为当前控件。

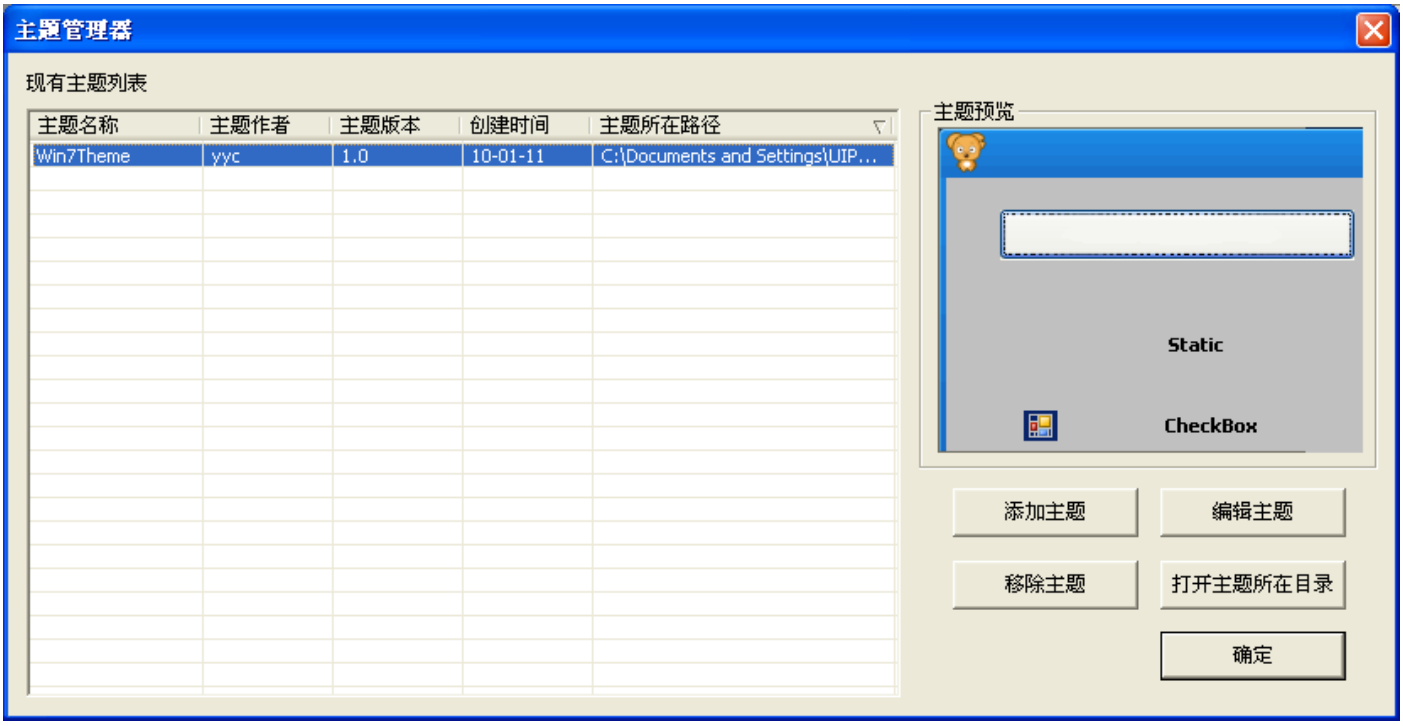
7） 不被拉伸区域

- 不被拉伸区域设定 2 个方向：横向与纵向。
- 在某个子区域中，如果他是横向的，则该部分的数值则表示与左边的间距和与右边的间距。
- 在某个子区域中，如果他是纵向的，则该部分的数值则表示与顶边的间距和与底边的间距。

8） 非客户区拉伸类型

- 有些子区域涉及到窗口非客户区域的拉伸，在这个地方可以进行设置。
- 非客户区域的拉伸有如下情况：Top、Left、Bottom、Right、TopLeft、TopRight、BottomLeft、BottomRight。
- 可以勾选 Use NCRegion 复选框来设置是否使用非客户区拉伸。

2.16 主题管理界面

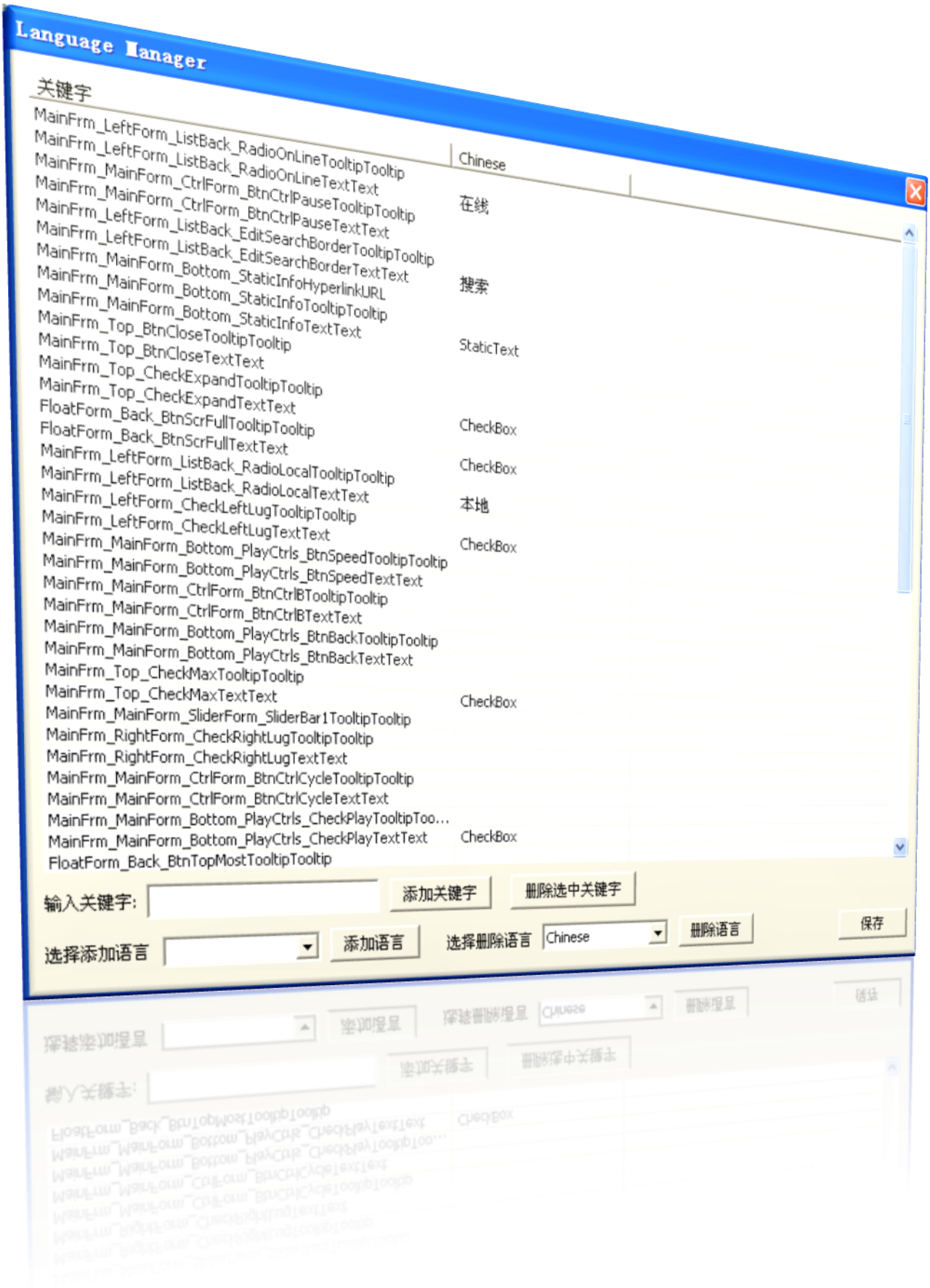


主题是 DirectUI 界面方案中很重要的一个概念。通过主题我们可以将重复的工作模板化。

该界面包括：主题列表、主题预览区、添加主题按钮、编辑主题按钮、移除主题按钮、打开主题所在目录按钮，确定按钮。

- 主题列表：列出 DirectUIBuilder 配置的所有的主题，包含作者、版本、创建时间、路径等信息。
- 主题预览区：选中列表中一个主题，预览区显示该主题的效果图。
- 添加主题：通过打开文件对话框选择磁盘中的一个主题文件，DirectUIBuilder 则记录下主题的文件路径。下次打开该列表中的主题还将存在。
- 编辑主题：选中列表中的一个主题，编辑该主题。DirectUIBuilder 将自动创建一个新的解决方案并将.dtheme 文件载入工程中，用户编辑后即保存至.dtheme 文件。
- 移除主题：选中列表中的一个主题，可以移除该主题。
- 打开主题所在目录：打开列表中选中主题所在的目录。

2.17 多语种管理界面



该界面包括关键字列表、增加关键字、删除关键字、添加语言、删除语言等功能。

- 关键字列表：

  第一列为关键字，一般有系统自动生成，名称由对象的绝对路径+属性的名称构成。

  第二列开始为语言列，一种语言一列，是对左边关键字的语言翻译。用户可以直接编辑选定关键字的某种语言的单元格。

- 增加关键字：除了系统自动生成的关键字以外，用户还可以添加自己的关键字，只要与列表中的关键字不重名即可。

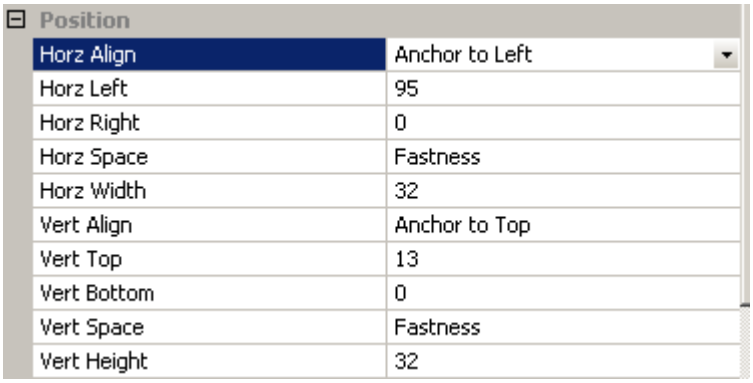
  一般与界面元素没有关系，同时程序中又需要用到的文字，并且该文字又需要支持多语种的情况才需要添加用户自己的关键字。

- 删除关键字：选中列表中的关键字，点击删除选中的关键字按钮即可删除。
- 添加语言：先选中可选语言列表中语言，点击添加语言按钮，即添加了一种语言，在关键字列表中会增加一列，显示语种的名称。
- 删除语言：先选中现有语言列表，点击删除语言按钮，即删除了选定的语言，在关键字列表中会将该列删除掉。

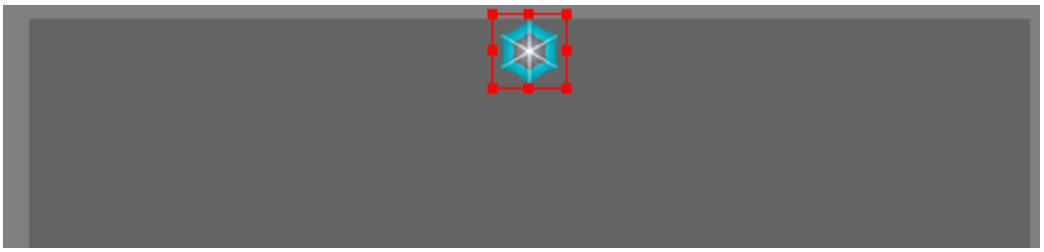
3. 控件布局介绍

不同于标准的控件，所有的 DirectUI 控件布局都由 DirectUI 平台进行管理，所有的控件遵循各自的布局规则进行位置的调整。

3.1 水平对齐方式



靠左对齐，居中对齐，靠右对齐



Position	
Horz Align	Anchor to Right
Horz Left	0

3.2 水平左端的偏移量

Position	
Horz Align	Anchor to Left
Horz Left	95
Horz Right	0
Horz Space	Fastness
Horz Width	32
Vert Align	Anchor to Top
Vert Top	13
Vert Bottom	0
Vert Space	Fastness
Vert Height	32

当设置为靠左对齐时有效，如设置为靠左对齐，距离左端 50 个像素

3.3 水平右端的偏移量

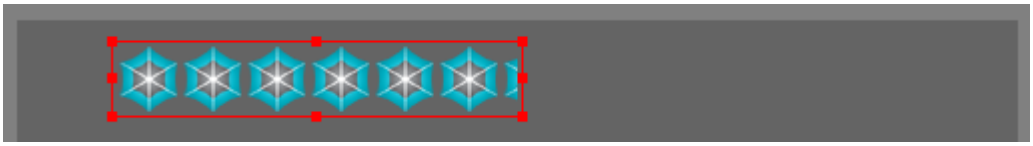
Position	
Horz Align	Anchor to Left
Horz Left	50
Horz Right	0
Horz Space	Fastness
Horz Width	32
Vert Align	Anchor to Top
Vert Top	13
Vert Bottom	0
Vert Space	Fastness
Vert Height	32

当设置为靠右对齐时有效，如设置为靠右对齐，距离右端 50 个像素

3.4 水平的宽度取值

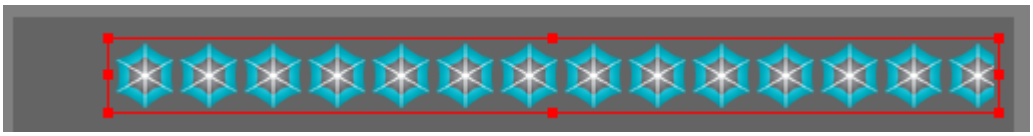
固定值，百分比，靠左，靠右

如设置为靠左对齐，距离左端 50 个像素，宽度为固定值，为 200 个像素



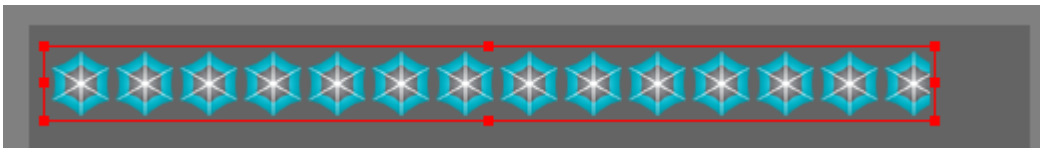
靠左，当设置为靠右对齐时有效

如设置为靠右对齐，距离右端 10 个像素，宽度取值为靠左，距离左端 50 个像素



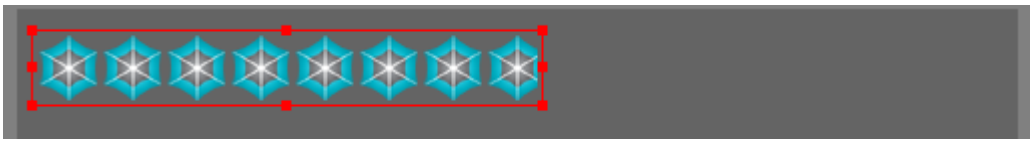
靠右，当设置为靠左对齐时有效

如设置为靠左对齐，距离左端 10 个像素，宽度取值为靠右，距离右端 50 个像素



百分比

如设置为 50%，则宽度为父控件宽度的 50%



3.5 垂直的设置与水平的设置类似

Position	
Horz Align	Anchor to Left
Horz Left	10
Horz Right	0
Horz Space	Right Width
Horz Width	50
Vert Align	Anchor to Top
Vert Top	13
Vert Bottom	0
Vert Space	Fastness
Vert Height	32

垂直对齐方式：靠上对齐、居中对齐、靠下对齐

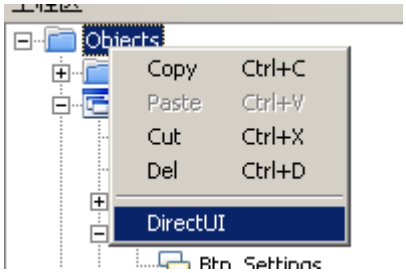
垂直的高度取值方式：靠上，靠下，百分比，固定值

4. KernelControls 控件及属性介绍

4.1 DirectUI 控件属性

DirectUI 控件是窗口控件，每一个窗口都需要独立建立一个 DirectUI 窗口进行对应。

Objects 节点下点击右键，选择 DirectUI 即可创建一个 DirectUI 对象。

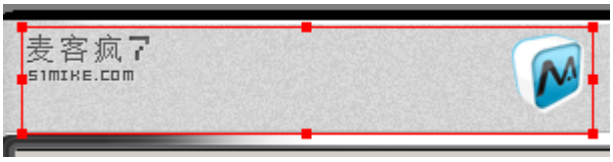


控件属性名称	属性类型	属性含义
Size【DirectUI 窗口大小】		
Width	Int	DirectUI 的窗口宽度
Height	Int	DirectUI 的高度
DragPosition【DirectUI 窗口拖拽范围】		
HorzAilgn	Option	DirectUI 可拖动水平范围的对齐方式 靠左、剧中、靠右
HorzLeft	Int	DirectUI 窗口可拖动水平范围向靠左距离
HorzRight	Int	DirectUI 窗口可拖动水平范围向靠右距离
HorzSpace	Option	DirectUI 窗口可拖动水平范围的取值规则 靠左，靠右，百分比，固定值
HorzWidth	Int	DirectUI 窗口可拖动水平范围的取值
VertAlign	Option	DirectUI 可拖动垂直范围的对齐方式 靠左、剧中、靠右
VertTop	Int	DirectUI 窗口可拖动垂直范围向靠左距离
VertBottom	Int	DirectUI 窗口可拖动垂直范围向靠右距离
VertSpace	Option	DirectUI 窗口可拖动垂直范围的取值规则 靠左，靠右，百分比，固定值
VertHeight	Int	DirectUI 窗口可拖动垂直范围的取值
Resize【DirectUI 窗口四周可拖拽改变大小的区域大小】		
Resizable	BOOL	窗口是否可改变大小
LeftWidth	Int	窗口左边框可托动范围
RightWidth	Int	窗口右边框可托动范围
TopHeight	Int	窗口顶部框可托动范围
BottomHeight	Int	窗口底部框可托动范围
Misc【其他的属性】		
Dragabled	BOOL	窗口是否可以拖动
DragfullWindow	BOOL	托动窗口时是否出现虚框
CanCopyRun	BOOL	窗口是否拥有多个实例 （例如：聊天窗口就是拥有多个实例，一个 DirectUI 皮肤需要被多个窗口使用，此时设置为 True）
MinWidth	Int	窗口最小的宽度，-1 为没有显示
MinHeight	Int	窗口最小的高度，-1 为没有显示
MaxWidth	Int	窗口最大的宽度，-1 为没有显示
MaxHeight	Int	窗口最大的高度，-1 为没有显示
TopMost	BOOL	窗口始终前端实现
MaxToScreenAll	BOOL	是大化的窗口的时候是否全屏（已过时）
SupportPerPixel	BOOL	窗体是否支持半透明窗口，半透明窗口不允许存在子窗口。

PopWindow	BOOL	是否是弹出窗口，非子窗口需要设置为 TRUE
ParentBlend	BOOL	是否父对象混合（已过时）
SetRgn	BOOL	是否通过图片来设置窗口 Rgn 区域
UseRgnInfo	BOOL	是否手动设置 Rgn 区域
SysMenu	BOOL	是否有系统按钮（已过时）
MinBox	BOOL	是否有最小化按钮（已过时）
MaxBox	BOOL	是否有最大化按钮（已过时）

4.2 UIForm 控件属性

UIForm 是一个可以绘制背景的容器控件



控件属性名称	属性类型	属性含义
Back【背景图片】		
DrawColor	BOOL	背景是否使用颜色绘制，False 则使用 ImageBase 绘制
Image	ImageBase	背景图片
Color	Colour	背景颜色
Mis【其他属性】		
DisposeMouse	BOOL	此控件是否截获鼠标消息不再向下转发，True 截获

4.3 Static 控件属性

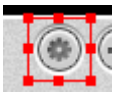
Static 为静态文本控件，用来显示窗口文字内容或者图片，并且支持滚动文字的功能。

控件属性名称	属性类型	属性含义
Back【背景图片】		
DisposeMouse	BOOL	是否放弃平台的鼠标消息
DrawColor	BOOL	绘图的时候用颜色还是用图片绘制
Active【激活状态属性】		
Image	BOOL	此控件是否截获鼠标消息不再向下转发，True 截获
Color	Color	激活状态下的颜色
TextStyle	TextStyle	激活状态下使用的文本样式
Text【显示文本内容】		

Text	Text	控件显示的文本内容
Text【自动调整大小功能】		
IsEnableAutoSize	BOOL	是否开启自动调整大小功能
MinTextHeight	Int	最小的文本高度
MaxTextHeight	Int	最大的文本高度
MinTextWidth	Int	最小的文本宽度
MaxTextWidth	Int	最大的文本宽度
Tooltip【Tooltip 文本】		
Text	Text	ToolTip 文本内容
EnableGetWord【当鼠标移动到控件上是否告知所指文本的内容，类似取词】		
EnableGetWord	BOOL	是否开启取词
GetWordType	Option	英文类型取词还是中文类型取词，英文类型根据空格和标点分割词组，中文根据一个 Uincode 来分割词组
HyperLink【Static 超文本样式】		
Hyperlink	BOOL	是否使用超链接样式，开启后 HyperlinkTextStyle 节点的属性方可生效。
URL	Text	用户点击此 static 后所跳转的网址
HyperlinkTextStyle【Static 超文本样式】		
Normal	TextStyle	正常状态下使用的文字样式
Press	TextStyle	按下状态下使用的文字样式
Hot	TextStyle	高亮状态下使用的文字样式
Disabled	TextStyle	禁用状态下使用的文字样式
ScrollText【滚动文本属性】		
Scroll	BOOL	是否使滚动文本
ScrollStep	Int	每一个时间周期的滚动距离
ScrollSpeed	Int	滚动速度，单位为毫秒
ScrollSpace	Int	文字滚动间距
Cursor【光标样式】		
Cursor	Cursor	鼠标移动到此控件上的光标样式

4.4 Button 控件属性

按钮控件，用来处理用户鼠标单击控件。



控件属性名称	属性类型	属性含义
Normal【正常状态属性】		
Image	ImageBase	正常状态下的图片
Graphics	ImageBase	正常状态下的图标
TextStyle	Text	正常状态下的文字

Press【按下状态属性】		
Image	ImageBase	按下状态下的图片
Graphics	ImageBase	按下状态下的图标
TextStyle	Text	按下状态下的文字
Disable【禁用状态属性】		
Image	ImageBase	禁用状态下的图片
Graphics	ImageBase	禁用状态下的图标
TextStyle	Text	禁用状态下的文字
Inactive【非激活状态属性，只有当此控件作为系统按钮时，属性方可有效，如关闭按钮】		
Image	ImageBase	非激活状态下的图片
Graphics	ImageBase	非激活状态下的图标
TextStyle	Text	非激活状态下的文字
Graphics【图标属性】		
UpDownMode	BOOL	水平居中还是垂直居中
RightAlign	BOOL	是否靠右对齐，否则靠左
X	Int	水平的偏移值
Y	Int	垂直的偏移值
HotKey【按钮的快捷键】		
Hotkey1	Hotkey	1 号热键
Hotkey2	Hotkey	2 号热键
IsBackSpace	BOOL	BackSpace 键是否允许作为热键
IsEnter	BOOL	IsEnter 键是否允许作为热键
Cursor【光标样式】		
Cursor	Cursor	鼠标移动到此控件上的光标样式

4.5 RadioBox 控件属性

单选控件，同一个 UIForm 下的单选控件将会自动建立为一个组。

控件属性名称	属性类型	属性含义
DrawMode【绘制的模式】		
DrawColor	BOOL	是用颜色还是使用图片绘制
Uncheck【未选中状态】		
Normal【正常状态】		
Back	ImageBase	未选中的正常状态下的背景图片
Color	colour	未选中的正常状态下的背景颜色
Box	ImageBase	未选中的正常状态下的图标背景
Graphics	ImageBase	未选中的正常状态下的图标前景
TextStyle	Text	未选中的正常状态下的文字风格

Hot【高亮状态】		
Back	ImageBase	未选中的高亮状态下的背景图片
Color	colour	未选中的高亮状态下的背景颜色
Box	ImageBase	未选中的高亮状态下的图标背景
Graphics	ImageBase	未选中的高亮状态下的图标前景
TextStyle	Text	未选中的高亮状态下的文字风格
Press【按下状态】		
Back	ImageBase	未选中的按下状态下的背景图片
Color	colour	未选中的按下状态下的背景颜色
Box	ImageBase	未选中的按下状态下的图标背景
Graphics	ImageBase	未选中的按下状态下的图标前景
TextStyle	Text	未选中的按下状态下的文字风格
Disable【禁用状态】		
Back	ImageBase	未选中的禁用状态下的背景图片
Color	colour	未选中的禁用状态下的背景颜色
Box	ImageBase	未选中的禁用状态下的图标背景
Graphics	ImageBase	未选中的禁用状态下的图标前景
TextStyle	Text	未选中的禁用状态下的文字风格
Check【选中状态】		
Normal【正常状态】		
Back	ImageBase	选中的正常状态下的背景图片
Color	colour	选中的正常状态下的背景颜色
Box	ImageBase	选中的正常状态下的图标背景
Graphics	ImageBase	选中的正常状态下的图标前景
TextStyle	Text	选中的正常状态下的文字风格
Hot【高亮状态】		
Back	ImageBase	选中的高亮状态下的背景图片
Color	colour	选中的高亮状态下的背景颜色
Box	ImageBase	选中的高亮状态下的图标背景
Graphics	ImageBase	选中的高亮状态下的图标前景
TextStyle	Text	选中的高亮状态下的文字风格
Press【按下状态】		
Back	ImageBase	选中的按下状态下的背景图片
Color	colour	选中的按下状态下的背景颜色
Box	ImageBase	选中的按下状态下的图标背景
Graphics	ImageBase	选中的按下状态下的图标前景
TextStyle	Text	选中的按下状态下的文字风格
Disable【禁用状态】		
Back	ImageBase	选中的禁用状态下的背景图片
Color	colour	选中的禁用状态下的背景颜色
Box	ImageBase	选中的禁用状态下的图标背景
Graphics	ImageBase	选中的禁用状态下的图标前景
TextStyle	Text	选中的禁用状态下的文字风格
Text【文本属性】		

Text	Text	Radio 的文本内容
ShowText	BOOL	是否显示文本
Tooltip【Tooltip 文本属性】		
Tooltip	Text	Tooltip 文本内容
Graphic【图标属性】		
UpdownMode	Int	水平对齐或者垂直对齐
x	Int	水平的偏移值
y	Int	垂直的偏移值
Width	Int	（已过时）
height	Int	（已过时）
Group【组的 ID】		
GroupID	Int	Radio 组的 Id
Cursor【光标样式】		
Cursor	Cursor	鼠标移动到此控件上的光标样式

4.6 CheckBox 控件属性

控件属性名称	属性类型	属性含义
Unchecked【未选中】		
Normal【正常状态】		
Image	Imagebase	未被选中的正常状态下左侧图标
TextStyle	Text	未被选中的正常状态下文本风格
Hot【高亮状态】		
Image	Imagebase	未被选中的高亮状态下左侧图标
TextStyle	Text	未被选中的高亮状态下文本风格
Press【按下状态】		
Image	Imagebase	未被选中的按下状态下左侧图标
TextStyle	Text	未被选中的按下状态下文本风格
Disable【正常状态】		
Image	Imagebase	未被选中的禁用状态下左侧图标
TextStyle	Text	未被选中的禁用状态下文本风格
Inactive【非激活状态】		

Image	Imagebase	未选中非激活状态下左侧图标
TextStyle	Text	未选中非激活状态下文本风格
Checked【选中】		
Normal【正常状态】		
Image	Imagebase	选中的正常状态下左侧图标
TextStyle	Text	选中的正常状态下文本风格
Hot【高亮状态】		
Image	Imagebase	选中的高亮状态下左侧图标
TextStyle	Text	选中的高亮状态下文本风格
Press【按下状态】		
Image	Imagebase	选中的按下状态下左侧图标
TextStyle	Text	选中的按下状态下文本风格
Disable【正常状态】		
Image	Imagebase	选中的禁用状态下左侧图标
TextStyle	Text	选中的禁用状态下文本风格
Inactive【非激活状态】		
Image	Imagebase	选中非激活状态下左侧图标
TextStyle	Text	选中非激活状态下文本风格
HalfCheck【半选】		
Normal【正常状态】		
Image	Imagebase	半选的正常状态下左侧图标
TextStyle	Text	半选的正常状态下文本风格
Hot【高亮状态】		
Image	Imagebase	半选的高亮状态下左侧图标
TextStyle	Text	半选的高亮状态下文本风格
Press【按下状态】		
Image	Imagebase	半选的按下状态下左侧图标
TextStyle	Text	半选的按下状态下文本风格
Disable【正常状态】		
Image	Imagebase	半选的禁用状态下左侧图标
TextStyle	Text	半选的禁用状态下文本风格
Inactive【非激活状态】		
Image	Imagebase	半选非激活状态下左侧图标
TextStyle	Text	半选非激活状态下文本风格
Cursor【光标属性】		
Cursor	Cursor	鼠标移动到此控件上的光标样式
PushLike【按钮样式】		
Value	BOOL	是否为按钮样式的 CheckBox

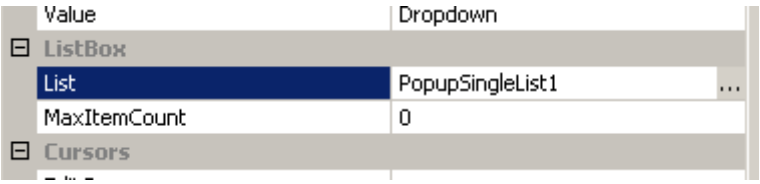
4.7 ComboBox 控件属性

下拉列表控件，一个下拉列表必须和一个下拉框绑定 PopupSingleList 后才可以正常使用，多个 Combox 可共用一个 PopupSingleList 控件。

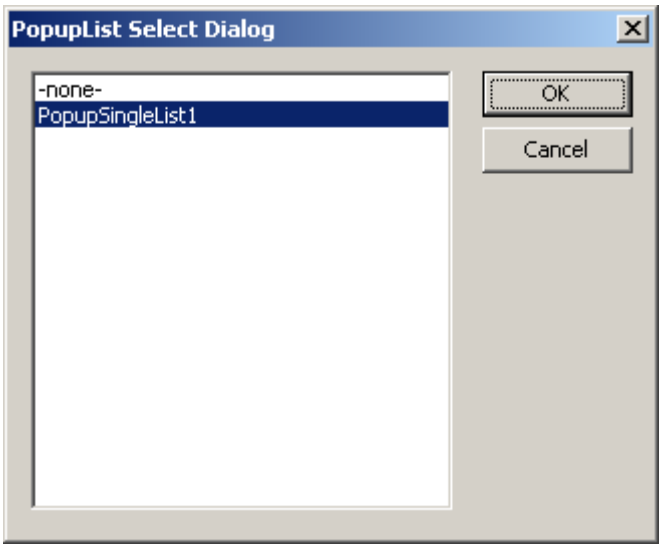
Combox

PopupSingleList

设置后效果



需要在样式里面选择使用某一个 PopupSingleList



控件属性名称	属性类型	属性含义
Border【边框背景】		
DrawColor	BOOl	是否使用颜色来绘制边框
ImageNormal	ImageBase	正常的背景图片
ColorNormal	Color	正常的边框颜色
ImagePress	ImageBase	按下时的背景图片
ColorPress	Color	按下时的背景颜色
ImageDisabled	ImageBase	禁用时的背景图片
ColorDisabled	Colour	禁用时的背景颜色
ImageHot	ImageBase	高亮时的背景图片
ColorHot	Color	高亮时的背景颜色
Button【下拉按钮】		
BackNormal	BOOl	正常的下拉按钮背景
GraphicsPress	ImageBase	按下的下拉按钮图标
BackPress	Colour	按下的下拉按钮背景

GraphicsPress	ImageBase	按下的下拉按钮图标
BackDisabled	Colour	禁用的下拉按钮背景
GraphicsDisabled	ImageBase	禁用的下拉按钮图标
BackHot	Colour	高亮的下拉按钮背景
GraphicsHot	ImageBase	高亮的下拉按钮图标
TextStyle【文本样式】		
Normal	TextStyle	正常状态下文本风格
Press	TextStyle	按下状态下文本风格
Disabled	TextStyle	禁用状态下文本风格
Hot	TextStyle	高亮状态下文本风格
ButtonSpace【文本与下拉按钮的间隔】		
Value	Int	文本与下拉按钮的间隔数值，单位像素
Tooltip【ToolTip】		
Edit	Text	鼠标移动到文本区域上的提示文本
Button	Text	鼠标移动到下拉按钮区域上的提示文本
DefaultText【默认文本】		
Value	Text	加载后文本区域默认显示的文本内容
ListBox【下拉列表样式】		
List	ControlPlugin	双击弹出选择框，选择一个下拉列表，PopupSingleList
MaxItemCount	Int	此下拉列表最大的 Item 个数
Cursor【光标】		
EditCursor	Cursor	文本区域上显示的光标样式
DropCursor	Cursor	下拉按钮上显示的光标样式

4.8 StarCtrl 控件属性

控件属性名称	属性类型	属性含义
Back【背景图片】		
Image	ImageBase	控件背景使用到的图片
Fore【前景图片】		
Image	ImageBase	控件前景使用到的图片
Misc【其他内容】		
Frames	Int	前景平分为多少个区域，即总共多少个星星
Pos	Int	当前显示的区域个数，即星星的个数
CanSetStar	BOOL	用户是否可通过鼠标来选择，即点击选择星星个数

4.9 ListView 控件属性

ListView 支持多种类型 Icon，SmallIcon，Report 和自定义

自定义类型是指使用一个 UIForm 作为一个 item 来使用，UIForm 中可以放置任意的控件。


Pre-S Questionnaire

1 Down a music from OVI

Description

Start Time 00:00:00 End Time 00:00:00

Ovserve:

Pre-S Questionnaire

Pre-S Questionnaire

Down a music form OVI

Pre-S Questionnaire

Pre-S Questionnaire

Down a music form OVI

Pre-S Questionnaire

Pre-S Questionnaire

Down a music form OVI

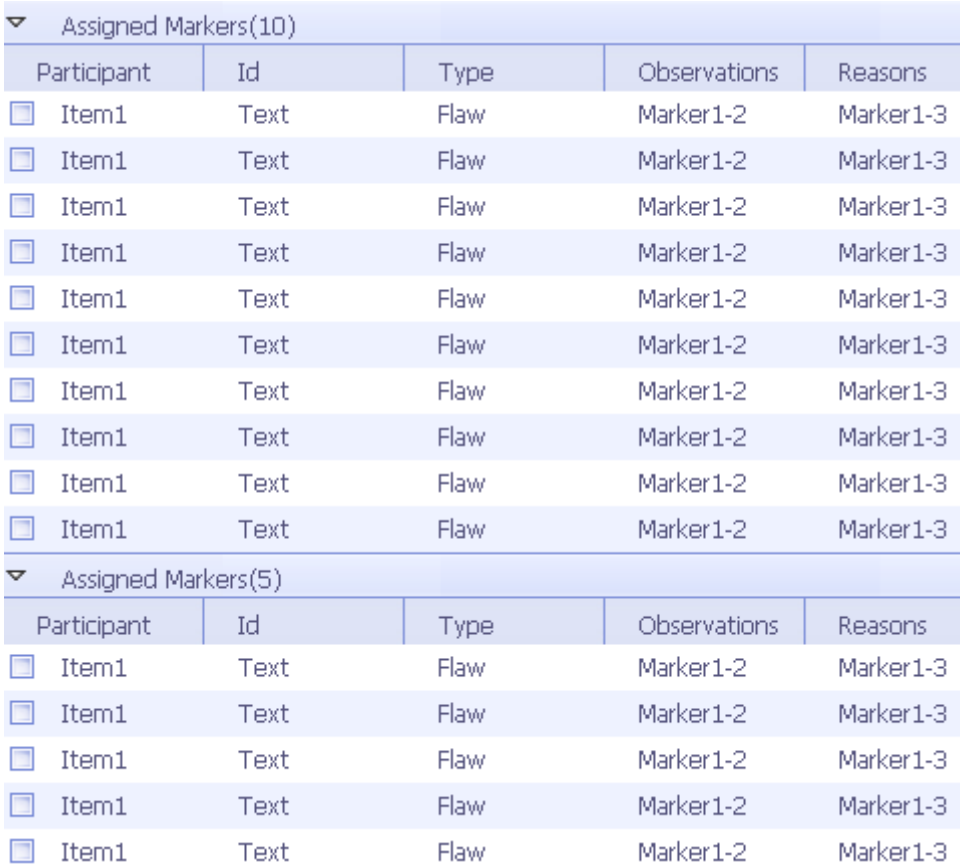
Pre-S Questionnaire

Pre-S Questionnaire

Down a music form OVI

Pre-S Questionnaire

自定义样式 ListView

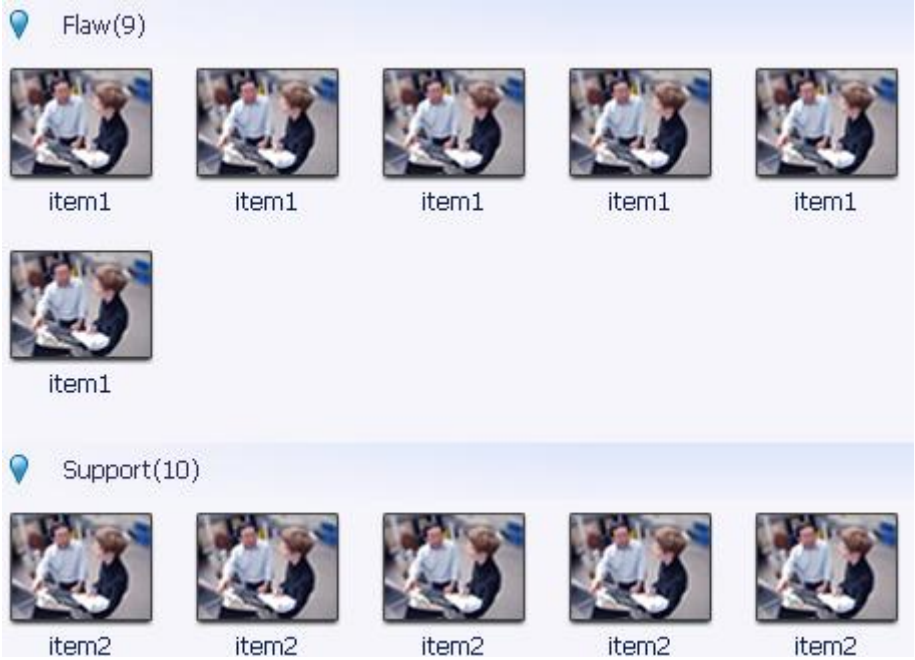

Assigned Markers(10)

	Participant	Id	Type	Observations	Reasons
☐	Item1	Text	Flaw	Marker1-2	Marker1-3
☐	Item1	Text	Flaw	Marker1-2	Marker1-3
☐	Item1	Text	Flaw	Marker1-2	Marker1-3
☐	Item1	Text	Flaw	Marker1-2	Marker1-3
☐	Item1	Text	Flaw	Marker1-2	Marker1-3
☐	Item1	Text	Flaw	Marker1-2	Marker1-3
☐	Item1	Text	Flaw	Marker1-2	Marker1-3
☐	Item1	Text	Flaw	Marker1-2	Marker1-3
☐	Item1	Text	Flaw	Marker1-2	Marker1-3
☐	Item1	Text	Flaw	Marker1-2	Marker1-3

Assigned Markers(5)

	Participant	Id	Type	Observations	Reasons
☐	Item1	Text	Flaw	Marker1-2	Marker1-3
☐	Item1	Text	Flaw	Marker1-2	Marker1-3
☐	Item1	Text	Flaw	Marker1-2	Marker1-3
☐	Item1	Text	Flaw	Marker1-2	Marker1-3
☐	Item1	Text	Flaw	Marker1-2	Marker1-3

Report 样式 ListView



Icon 样式 listview

smallIcon 和 Icon 样式类似，唯一的区别是文字处在图标的右边。

控件属性名称	属性类型	属性含义
ListType【Listview 的类型】		
Type	Options	ListView 的类型，Icon 类型，SmallIcon 类型，Report 类型，自定义类型
Back【背景图片】		
DrawColor	BOOL	是否使用颜色来绘制背景，否则使用图片
Image	ImageBase	ListView 的背景图片
Color	colour	ListView 的背景颜色
DrawGrid	BOOL	是否画表格线（只有 Report 样式下有效）
SupportMultiColor	BOOL	是否支持间隔颜色（只有 Report 样式下有效）
SecColor	FillColour	间隔色，使用 Color 和 SecColor 的颜色交替绘制（只有 Report 样式下有效）
Group【组属性】		
SupportGroup	BOOL	是否支持 Group
CanExpandGroup	BOOL	Group 是否支持展收缩
ExPandbyIcon	BOOL	Group 是否通过点击组前面的图标来响应收缩，否则点击组所在的一整行响应收缩
CaptionHeight	Int	Group 组的高度
ArrawIcon【组箭头属性】		
AlignRight	BOOL	是否靠右对齐，否则靠左对齐
IconOffsetX	Int	箭头图标的水平偏移量
UnpandIconImage	ImageBase	组收缩时显示的图标
ExexpandIconImage	ImageBase	组展开时显示的图标
CaptionImages【组背景】		
Normal	ImageBase	组正常状态下的背景

Hot	ImageBase	组高亮状态下的背景
Press	ImageBase	组按下状态下的背景
Disabled	ImageBase	组禁用状态下的背景
CaptionImages【组背景】		
Normal	TextStely	组正常状态下使用的文字样式
Hot	TextStely	组高亮状态下使用的文字样式
Press	TextStely	组按下状态下使用的文字样式
Disabled	TextStely	组禁用状态下使用的文字样式
CaptionImages【组右侧可以设置一个按钮，点击此按钮可以关闭此组】		
ButtonOffset	Int	关闭图标的水平偏移量
Width	Int	关闭图标的宽度
Height	Int	关闭图标的高度
Normal	ImageBase	关闭图标的正常状态图片
Hot	ImageBase	关闭图标的高亮状态图片
Press	ImageBase	关闭图标的按下状态图片
Disabled	ImageBase	关闭图标的禁用状态图片
Item【List Item 属性】		
ListModel	CtrlPlugin	选择一个 UIForm 作为一个 Item 的模板
DrawColor	BOOL	是否使用颜色来绘制 Item 的背景
Height	Int	Item 的高度
Width	Int	Item 的宽度
Normal【Item 的正常状态属性】		
Image	ImageBase	Item 正常状态下的背景图片
Color	colour	Item 正常状态下的背景颜色
TextStyle	Text	Item 正常状态下的文本样式
Press【Item 的按下状态属性】		
Image	ImageBase	Item 按下状态下的背景图片
Color	colour	Item 按下状态下的背景颜色
TextStyle	Text	Item 按下状态下的文本样式
Normal【Item 高亮状态属性】		
Image	ImageBase	Item 高亮状态下的背景图片
Color	colour	Item 高亮状态下的背景颜色
TextStyle	Text	Item 高亮状态下的文本样式
Normal【Item 禁用状态属性】		
Image	ImageBase	Item 禁用状态下的背景图片
Color	colour	Item 禁用状态下的背景颜色
TextStyle	Text	Item 禁用状态下的文本样式
UnitItem【子单元格属性，此属性仅 Report 样式有效】		
SupporMultiRow	BOOL	子单元格是否支持多行文本
StaticItem【子单元格中静态文本属性，此属性仅 Report 样式有效】		
Normal	TextStely	子单元格中静态文本的正常状态下的文本样式
Hot	TextStely	子单元格中静态文本的正常状态下的文本样式
Press	TextStely	子单元格中静态文本的正常状态下的文本样式
Disabled	TextStely	子单元格中静态文本的正常状态下的文本样式
UrlItem【子单元格中超链接文本属性，此属性仅 Report 样式有效】		

Normal	TextStely	子单元格中超链接文本的正常状态下的文本样式
Hot	TextStely	子单元格中超链接文本的正常状态下的文本样式
Press	TextStely	子单元格中超链接文本的正常状态下的文本样式
Disabled	TextStely	子单元格中超链接文本的正常状态下的文本样式
Misc【其他设置】		
SupporDrag	BOOL	是否支持拖拽交互 Item 的位置（SmallIcon 模式和 List 模式有效）
DragByPoseMsg	BOOL	拖拽后抛出消息不立即改变 Item 位置（SmallIcon 模式和 List 模式有效）
SupportMultiSel	BOOL	是否支持 Item 的多选
ItemSpace	Int	Item 的之间的间隔
LeftMargin	Int	Item 离左边的间隔
RighMargin	Int	Item 离右边的间隔
LineColor	ImageBase	DrawGrid 中线的颜色，绘制表格线的颜色
SelColumnColor	color	选中列的颜色（仅 Report 样式有效）
AutoResizeItemHeight	BOOL	是否自动调整 Item 的高度（仅 Report 样式有效）
CheckOffset	Int	Report 样式下，Check 图标的靠左偏移量(仅 Report 样式有效)
ReportFullRowHot	BOOL	Report 样式下是否整行高亮，否则仅一个单元格高亮
SupportVisualList	BOOL	是否使用续表来提高 ListView 的显示效率
CustomClickOnly	BOOL	是否分离自定义点击消息和 Listview 的 Item 点击消息，默认情况下会同时抛出这两个消息（仅自定义模式下有效）
ScrollBarInfo【滚动条属性】		
ScroLLsize	Int	滚动条的宽度
leftScroll	BOOL	滚动条是否靠左，否者靠右
UseScrollBar	BOOL	是否使用滚动条
Icon【Icon 属性，仅 Icon 和 SmallIcon 模式下有效】		
AutoSize	BOOL	根据 Icon 图片自动调整大小
AutoWidth	Int	Icon 的宽度
AutoHeight	Int	Icon 的高度
OffsetX	Int	Icon 的水平偏移量
OFFsetY	Int	Icon 的垂直偏移量
CanRightBottomIcon	BOOL	Icon 模式下 Icon 是否处在文本下方，默认为上方 Small 模式下 Icon 是否处在文本右边，默认为左边
CheckBox【Report 样式下，Item 前面的 CheckBox 样式】		
UncheckedImage【未选中显示效果】		
Normal	ImageBase	未选中正常状态下的图标
Hot	ImageBase	未选中高亮状态下的图标
Press	ImageBase	未选中按下状态下的图标
Disable	ImageBase	未选中禁用状态下的图标
UncheckedImage【选中显示效果】		
Normal	ImageBase	选中正常状态下的图标
Hot	ImageBase	选中高亮状态下的图标
Press	ImageBase	选中按下状态下的图标
Disable	ImageBase	选中禁用状态下的图标
Cursor【光标样式】		

DragCursor	Cursor	拖拽 Item 时显示的光标
Cursor	Cursor	移动移动到控件上显示的光标

4.10 Animate 控件属性

动画控件，目前仅仅支持帧动画。

需要美术制作如上所示的帧图片，然后作为 ImageBase 设置给动画控件。

控件属性名称	属性类型	属性含义
ImageMode【动画类型】		
Mode	Options	动画控件的类型，仅 Frames 模式有效，即帧动画类型有效
Gif【Gif 动画类型，此属性目前不支持】		
Image	ImageBase	（无效）
ImageFrames【帧动画类型】		
Image	ImageBase	帧动画图片
FramesCount	Int	帧动画图片分割数量
Speed	Int	播放速度，单位毫秒
Auto【自动播放属性】		
AutoStart	BOOL	控件加载后自动播放
AutoStop	BOOL	控件动画播放一遍后自动停止
Rgn【根据图片计算 RGN 区域，此属性无效】		
SetRgn	BOOL	（无效）

4.11 LogoObj 控件属性

头像控件

控件属性名称	属性类型	属性含义
BorderImage【背景图片】		
Image	ImageBase	背景图片
BorderSize【中央图片和四周的间隔】		
Left	Int	中央图片与左端的距离

Right	Int	中央图片与右端的距离
Top	Int	中央图片与顶端的距离
Bottom	Int	中央图片与低端的距离
Cursor【光标属性】		
Cursor	Cursor	鼠标移动到此控件上的光标样式

4.12 Tab 控件属性

标签页控件

控件属性名称	属性类型	属性含义
Style【Tab 样式】		
Value	Option	Tab 的显示样式，Item 方向向上，向左，向右，向下
Background【Tab 背景属性】		
DrawBackColor	BOOL	是否使用颜色来绘制背景，否则使用图片
Normal【Tab 背景属性】		
Color	Colour	Tab 背景颜色
PageColor	Colour	（无效）
Image	ImageBase	Tab 背景图片
PageImage	ImageBase	（无效）
Disable【Tab 禁用状态背景属性】		
Color	Colour	Tab 禁用状态背景颜色
PageColor	Colour	（无效）
Image	ImageBase	Tab 禁用状态背景图片
PageImage	ImageBase	（无效）
Border【Item 与背景的间距】		
Left	Int	Item 与控件左端的间距
Top	Int	Item 与控件顶端的间距
Right	Int	Item 与控件右端的间距
Bottom	Int	Item 与控件低端的间距
Item【TabItem 的属性】		
OffsetX	Int	Item 里面的图标水平偏移量
OffsetY	Int	Item 里面的图标垂直偏移量
Space	Int	两个 Item 之间的间隔
AutoResize	BooL	Item 是否根据文本的宽度来自动计算宽度
Width	Int	Item 的宽度，设置固定值

Height	Int	Item 的高度，设置固定值
Tooltip	BOOL	鼠标移动到 Item 上是否显示 Tooltip
SpaceWithButton	Int	Item 与滚动按钮的间距
TabPage	Dialog	Item 编辑对话框，用于编辑 Tab 页的静态 Item
Normal【Item 正常状态的属性】		
Image	ImageBase	Tab 项正常状态的背景
TextStyle	Text	Tab 项正常状态文本风格
Disable【Item 禁用状态的属性】		
Image	ImageBase	Tab 项禁用状态的背景
TextStyle	Text	Tab 项禁用状态文本风格
Hot【Item 正常状态的属性】		
Image	ImageBase	Tab 项高亮状态的背景
TextStyle	Text	Tab 项高亮状态文本风格
Press【Item 按下状态的属性】		
Image	ImageBase	Tab 项按下状态的背景
TextStyle	Text	Tab 项按下状态文本风格
ItemCloseButton【每一个 Item 可拥有一个关闭按钮，关闭按钮属性】		
ItemHaveCloseBtn	BOOL	支持 Item 是否有关闭按钮
Width	Int	按钮宽度
Height	Int	按钮高度
RightOffsetX	Int	水平方向靠右偏移，按钮默认是靠右布局的
OffsetY	Int	垂直方向的偏移
SpaceWithItem	Int	与 Item 文本的间隔
Normal.Image	ImageBase	正常状态图片
Disabled.Image	ImageBase	禁用状态图片
Hot.Image	ImageBase	高亮状态图片
Press.Image	ImageBase	按下状态图片
Focus.Image	ImageBase	焦点状态图片
ArrawButton【Tab 中如果 Item 数量超过可显示的数量时会出现滚动按钮】		
IsArrowBtn	BOOL	是否显示滚动按钮
Width	Int	滚动按钮的宽度
Height	Int	滚动按钮的高度
Space	Int	向前和向后两个滚动按钮的间隔
BackBtnOffsetX	Int	向前滚动按钮水平偏移
BackBtnOffsetY	Int	向前滚动按钮垂直偏移
NextBtnOffsetX	Int	向后滚动按钮水平偏移
NextBtnOffsetY	Int	向后滚动按钮垂直偏移
Normal【滚动按钮正常状态属性】		
BackBtnImage	ImageBase	向前滚动按钮正常状态图片
NextBtnImage	ImageBase	向后滚动按钮正常状态图片
Disable【滚动按钮禁用状态属性】		
BackBtnImage	ImageBase	向前滚动按钮禁用状态图片
NextBtnImage	ImageBase	向后滚动按钮禁用状态图片
Hot【滚动按钮高亮状态属性】		
BackBtnImage	ImageBase	向前滚动按钮高亮状态图片

NextBtnImage	ImageBase	向后滚动按钮高亮状态图片
Press【滚动按钮按下状态属性】		
BackBtnImage	ImageBase	向前滚动按钮按下状态图片
NextBtnImage	ImageBase	向后滚动按钮按下状态图片
Press【滚动按钮获得焦点状态属性】（此属性无效）		
BackBtnImage	ImageBase	向前滚动按钮获得焦点状态图片（此属性无效）
NextBtnImage	ImageBase	向后滚动按钮获得焦点状态图片（此属性无效）
CloseButton【Tab 在最后侧可以放置一个关闭按钮，用来关闭当前的 Item】		
HaveCloseBtn	BOOL	Tab 控件最右侧是否有关闭按钮
Normal.Image	ImageBase	关闭按钮正常状态图片
Disabled.Image	ImageBase	关闭按钮禁用状态图片
Hot.Image	ImageBase	关闭按钮高亮状态图片
Press.Image	ImageBase	关闭按钮按下状态图片
Size【关闭按钮的大小】		
Width	Int	Tab 控件宽度大小
Height	Int	Tab 控件高度大小
Offset【关闭按钮位置的偏移量】		
x	Int	水平滚动的数值
Y	Int	垂直滚动的数值
Misc【其他设置】		
AutoResize	BOOL	根据 Item 的数量和宽度来自动调整控件的宽度
IsMouseInRgn	BOOL	无效
ScrollPerStep	Int	点击滚动按钮，滚动 Item 时每次的距离，单位像素
ScrollSpeed	Int	点击滚动按钮，滚动 Item 的速度，单位毫秒
currentpage	Int	当前显示的 Item（Builder 内预览属性）
TabHeight	Int	无效
ScrollshowfullItem	BOOL	当有多个 Item 时，是否禁止出现半个 Item 的现象
ItemDragable	BOOL	是否可拖拽调整 Item 的顺序

4.13 SplitterBar 控件属性

分割条控件，通过分割条我们可以调整两个 UIForm 的大小

属性区中分别填写两个 UIForm 的名称，并且两个 UIForm 的布局必须遵循以下的规则：

如果是水平方向上，一个 UIForm 靠左宽度固定值，另一个距离左端和宽度靠右

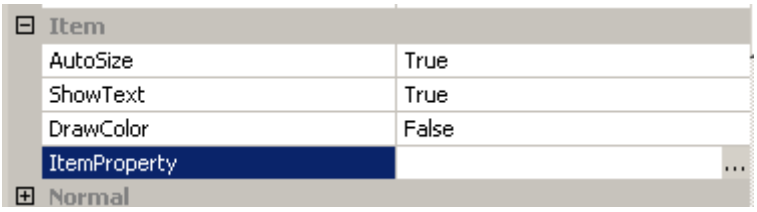
如果是垂直方向上，一个 UIForm 靠上高度固定值，另一个距离上端和高度靠下

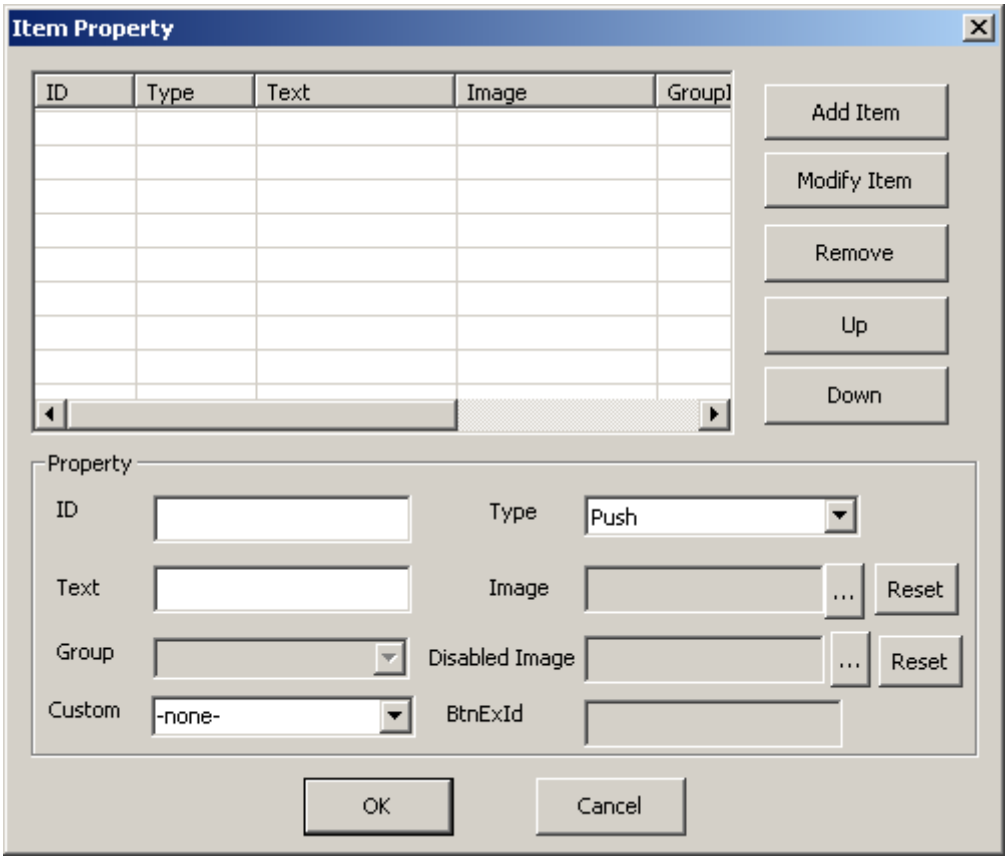
控件属性名称	属性类型	属性含义
Image【背景图片】		
Normal【正常状态属性】		
Image	Imagebase	正常状态的背景图片
Graphics	Imagebase	正常状态的图标，默认水平和垂直居中显示
Press【按下状态属性】		

Image	Imagebase	按下状态的背景图片
Graphics	Imagebase	按下状态的图标，默认水平和垂直居中显示
Disable【禁用状态属性】		
Image	Imagebase	禁用状态的背景图片
Graphics	Imagebase	禁用状态的图标，默认水平和垂直居中显示
Hot【高亮状态属性】		
Image	Imagebase	高亮状态的背景图片
Graphics	Imagebase	高亮状态的图标，默认水平和垂直居中显示
Hot【其他属性】		
Graphlenght	Int	图片的高度
IsHorz	BOOL	是否是水平方向的分隔条
VirtualLinee	BOOL	是否使用虚线表示拖拽的位置
BarPressColor	Colour	无效
TopMin	Int	第一分割区域的最小值
BottomMin	Int	第二分割区域的最小值
UpCtrls	Text	第一分割 UIForm 的名称
DownCtrls	Text	第二分割 UIForm 的名称
Cursor【光标属性】		
Cursor	Cursor	拖拽时的光标

4.14 ToolBar 控件属性

工具栏控件，通过 ItemProperty 属性我们可以静态设置 toolbar 的 item





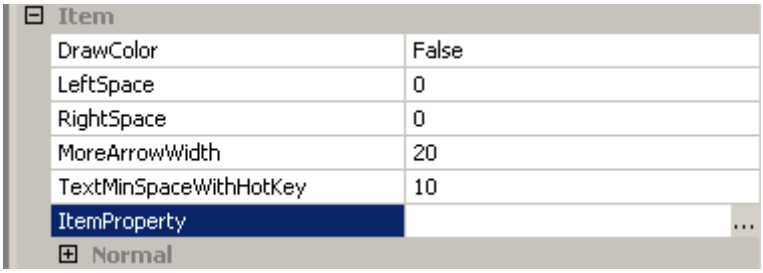
控件属性名称	属性类型	属性含义
Back【背景图片】		
DrawColor	BOOL	是否使用颜色来绘制控件的背景，否则使用图片
Image	ImageBase	背景的图片
Color	colour	背景的颜色
Item【背景图片】		
AutoSize	BOOL	Item 自动调整大小
ShowText	Text	显示文字
DrawColor	colour	是否使用颜色来绘制 Item 的背景
ItemProperty	Dialog	Item 编辑窗口，用于添加、修改和删除 ToolBar Item
Normal【Item 正常状态】		
ItemImage	ImageBase	Item 正常状态的背景图片
ItemExImage	ImageBase	扩展按钮类型 Item 正常状态的的图片
ItemColor	Colour	Item 正常状态的背景颜色
TextStyle	Text	Item 正常状态的文字样式
Hot【Item 高亮状态】		
ItemImage	ImageBase	Item 高亮状态的背景图片
ItemExImage	ImageBase	扩展按钮类型 Item 高亮状态的的图片
ItemColor	Colour	Item 高亮状态的背景颜色
TextStyle	Text	Item 高亮状态的文字样式
Press【Item 按下状态】		
ItemImage	ImageBase	Item 按下状态的背景图片
ItemExImage	ImageBase	扩展按钮类型 Item 按下状态的的图片
ItemColor	Colour	Item 按下状态的背景颜色

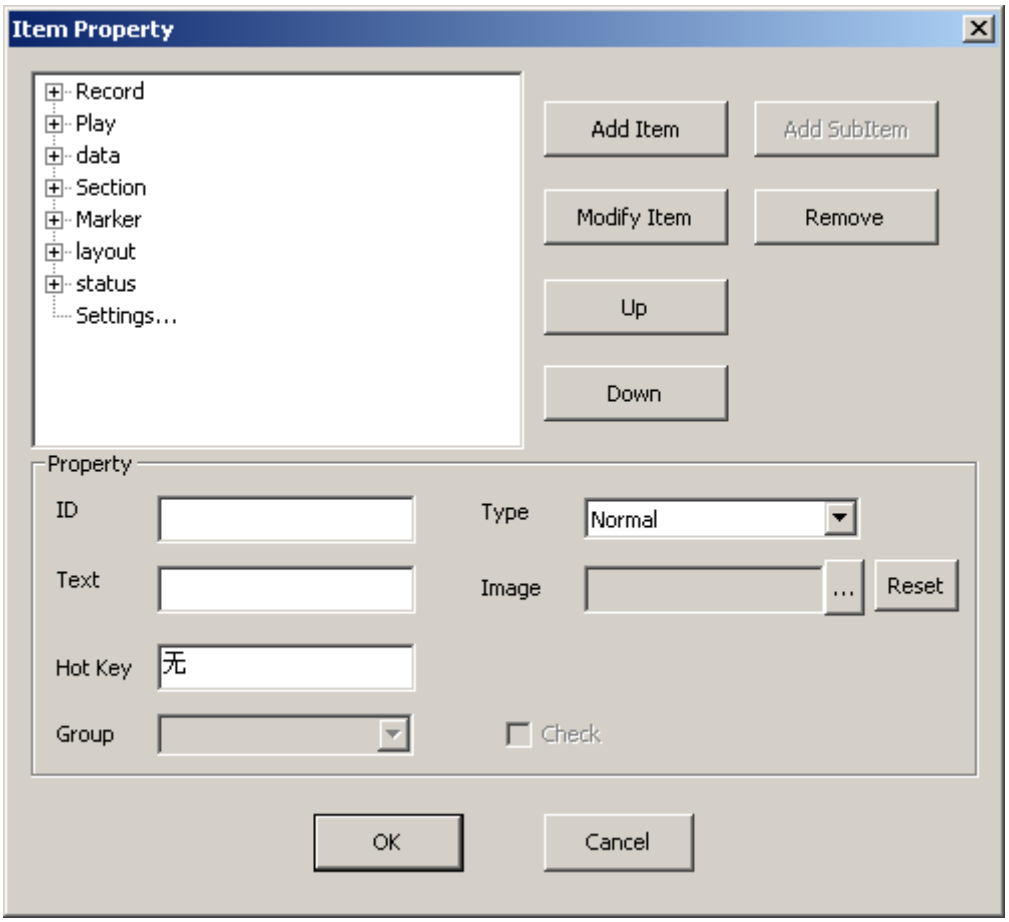
TextStyle	Text	Item 按下状态的文字样式
Disable【Item 禁用状态】		
ItemImage	ImageBase	Item 禁用状态的背景图片
ItemExImage	ImageBase	扩展按钮类型 Item 禁用状态的的图片
ItemColor	Colour	Item 禁用状态的背景颜色
TextStyle	Text	Item 禁用状态的文字样式
Selected【Check 类型 Item 选中状态】		
ItemImage	ImageBase	Check 类型 Item 选中状态的背景图片
ItemExImage	ImageBase	无效
ItemColor	Colour	Check 类型 Item 选中状态的背景颜色
TextStyle	Text	Check 类型 Item 选中状态的文字样式
Icon【Item 上图标属性】		
OffsetX	Int	图标水平方向偏移量
OffsetY	Int	图标垂直方向偏移量
IconWidth	Int	图标的宽度
IconHeight	Int	图标的高度
IconAutoSize	BOOL	图标是否自动调整大小
HorzWithText	BOOL	文字是否处在图标的右侧，否则为下方
Gripper【Gripper 属性】		
Image	ImageBase	Gripper 的背景
Size	Int	Gripper 的大小
leftSpace	Int	Gripper 靠左边的间距
TopSpace	Int	Gripper 靠上边的间距
BottomSpace	Int	Gripper 靠底边的间距
Separator【分隔符属性】		
DrawColor	BOOL	分隔符背景是否用颜色绘制，否则使用图片
Image	ImageBase	分隔符背景图片
Color	colour	分割符背景颜色
Size	Int	分割符的宽度
SpaceWithItem	Int	分隔符和 Item 的间距
MoreButton【当 Item 数量超过显示范围时会出现 MoreButton】		
NormalImage	ImageBase	显示更多按钮正常状态的图片
HotImage	ImageBase	显示更多按钮高亮状态的图片
PressImage	ImageBase	显示更多按钮按下状态的图片
DisabledImage	ImageBase	显示更多按钮正禁用状态的图片
SelectedImage	ImageBase	显示更多挑选的图片
Size	Int	更多的按钮的宽度
RightSpace	Int	默认更多按钮处在控件的右侧，与右侧的间距
BottomSpace	Int	与控件下方的间距
Misc【其他属性】		
Horz	BOOL	是否为水平方向的 ToolBar
ItemLeftTopOffset	Int	Item 靠左上偏移量
ItemHeight	Int	Item 的高度
ItemWidth	Int	Item 的宽度
ItemSpace	Int	Item 之间的间距

ButtonExWidth	Int	扩展按钮宽度
MorePage【点击 MoreButton 后弹出 Item 列表窗口，此窗口属性】		
Image	ImageBase	Item 列表窗口的背景
PageWidth	Int	Item 列表窗口的宽度

4.15 PopupMenu 控件属性

弹出菜单控件，通过 ItemProperty 属性可以设置静态的 item





控件属性名称	属性类型	属性含义
Size【弹出菜单的大小】		
Width	Int	菜单的宽度
Height	Int	菜单的高度
Back【弹出菜单的背景】		
DrawColor	BOOL	是否使用颜色来绘制背景，否则使用图片
Image	ImageBase	背景图片
Color	colour	背景颜色
IconRegion【弹出菜单的图标区域】		
Width	Int	图标区域的宽度
DrawColor	BOOL	是否用颜色来绘制图标区域，否则使用图片
Image	ImageBase	图标区域背景图片
Color	colour	图标区域背景颜色
NcBorder【弹出菜单 Item 与四周的间距】		
LeftBorder	Int	左边框的间距
RightBorder	Int	右边框的间距
TopBorder	Int	上边框的间距
BottomBorder	Int	底边框的间距
Item【弹出菜单 Item】		
DrawColor	BOOL	Item 是否用颜色来绘制背景，否则使用图片
LeftSpace	Int	Item 离左边的间距
RightSpace	Int	Item 离右边的间距
MoreArrowWidth	Int	子菜单箭头的宽度

TextMinSpaceWithHotKey	Int	文字与热键之间的间距
ItemProPerty	Dialog	弹出菜单 Item 编辑对话框，用于添加、删除、修改菜单项
Normal【Item 正常状态属性】		
ItemImage	ImageBase	Item 正常状态下的背景图片
ItemColor	colour	Item 正常状态下的背景颜色
TextStyle	Text	Item 正常状态下的文字风格
MoreArrow	ImageBase	Item 正常状态下更多箭头图标
Hot【Item 高亮状态属性】		
ItemImage	ImageBase	Item 高亮状态下的背景图片
ItemColor	colour	Item 高亮状态下的背景颜色
TextStyle	Text	Item 高亮状态下的文字风格
MoreArrow	ImageBase	Item 高亮状态下更多箭头图标
Press【Item 按下状态属性】		
ItemImage	ImageBase	Item 按下状态下的背景图片
ItemColor	colour	Item 按下状态下的背景颜色
TextStyle	Text	Item 按下状态下的文字风格
MoreArrow	ImageBase	Item 按下状态下更多箭头图标
Disable【Item 禁用状态属性】		
ItemImage	ImageBase	Item 禁用状态下的背景图片
ItemColor	colour	Item 禁用状态下的背景颜色
TextStyle	Text	Item 禁用状态下的文字风格
MoreArrow	ImageBase	Item 禁用状态下更多箭头图标
Selected【Check 类型 Item 选中状态属性】		
ItemImage	ImageBase	Check 类型 Item 选中状态下的背景图片
ItemColor	colour	Check 类型 Item 选中状态下的背景颜色
TextStyle	Text	Check 类型 Item 选中状态下的文字风格
MoreArrow	ImageBase	Check 类型 Item 选中状态下更多箭头图标
Misc【其他属性】		
SupportPerPicxel	BOOL	是否支持半透明菜单效果
Opacity	Int	透明度（0-255 之间）
ItemHeight	Int	Item 的高度
ItemSpace	Int	Item 之间的间距
MaxHeight	Int	菜单的最大高度
MaxWidth	Int	菜单的最大宽度
MinWidth	Int	菜单的最小宽度
Separator【分隔条类型 Item 属性】		
DrawColor	BOOL	分割线是否使用颜色来绘制背景，否则使用图片
Image	ImageBase	分割线使用的背景图片
Color	FillColor	分割线使用的背景颜色
Height	Int	分割线的高度
OffsetLeft	Int	分割线的水平偏移
OffsetRight	Int	分割线的垂直偏移
SpaceWithItem	Int	分隔条与上下 Item 的间距
CheckBox【Check 类型 Item 属性】		

Normal	ImageBase	Check 类型 Item 图标的正常状态图片
Hot	ImageBase	Check 类型 Item 图标的高亮状态图片
Press	ImageBase	Check 类型 Item 图标的按下状态图片
Disabled	ImageBase	Check 类型 Item 图标的禁用状态图片
Selected	ImageBase	Check 类型 Item 图标的选中状态图片
RadioImage【Radio 类型 Item 属性】		
Normal	ImageBase	Radio 类型 Item 图标的正常状态图片
Hot	ImageBase	Radio 类型 Item 图标的高亮状态图片
Press	ImageBase	Radio 类型 Item 图标的按下状态图片
Disabled	ImageBase	Radio 类型 Item 图标的禁用状态图片
Selected	ImageBase	Radio 类型 Item 图标的选中状态图片
Icon【Item 图标的属性】		
AutoSize	BOOL	图标是否根据图片大小自动调整大小
AutoWith	Int	图标的宽度
AutoHeight	Int	图标的高度
AutoCenter	BOOL	图标是否居中
OffsetX	Int	图标水平偏移量
OffsetY	Int	图标垂直偏移量
Animate【菜单弹出动画】		
AniType	Combobox	动画类型，Normal 正常弹出，Slide 从上向下滑出，Blend 逐渐显示出来
Speed	Int	动画速度，单位毫秒
ArrowButton（无效）		
OffsetX	Int	（无效）
OffsetY	Int	（无效）
Normal（无效）		
Image	ImageBase	（无效）
color	Colour	（无效）
Hot（无效）		
Image	ImageBase	（无效）
color	Colour	（无效）
Disable（无效）		
Image	ImageBase	（无效）
color	Colour	（无效）
Selected（无效）		
Image	ImageBase	（无效）
color	Colour	（无效）
SubMenu（子菜单属性）		
TimeDelay	Int	子菜单出现的时间延时
HorzSpace	Int	子菜单与父菜单之间的水平间距

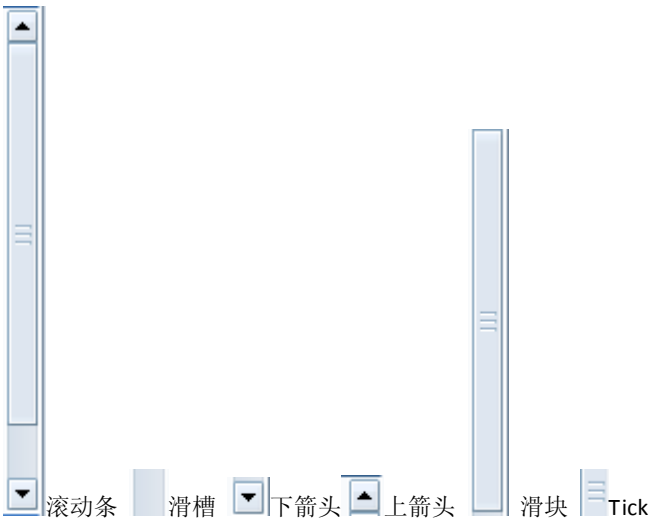
4.16 Menubar 菜单栏控件属性

菜单栏控件，可以和 DirectUI 弹出菜单绑定或者标准菜单绑定。

控件属性名称	属性类型	属性含义
Backgounrd【背景属性】		
DrawColor	BOOL	是否使用颜色绘制背景，否则使用图片
ActiveColor	FillColour	激活状态下的背景颜色
ActiveImage	FillColour	激活状态下的图片
InactiveColor	FillColour	非激活状态下的颜色
InactiveImage	FillColour	非激活状态下的图片
Item【菜单项属性】		
ItemHeight	Int	Item 的高度
ItemSpace	Int	Item 之间的间距
MeunBarHeight	Int	（无效）
OffsetLeft	Int	Item 与控件左端的间隔
OffsetRight	Int	Item 与控件右端的间隔
OffsetTop	Int	Item 与控件上端的间隔
Normal【菜单项正常状态属性】		
Image	ImageBase	正常状态下的背景图片
TextStyle	TextStyle	正常状态下的文字样式
Selected【菜单项选中状态属性】		
Image	ImageBase	选中状态下的背景图片
TextStyle	TextStyle	选中状态下的文字样式
Disable【菜单项禁用状态属性】		
Image	ImageBase	禁用状态下的背景图片
TextStyle	TextStyle	禁用状态下的文字样式
Hot【菜单项高亮状态属性】		
Image	ImageBase	正常状态下的背景图片
TextStyle	TextStyle	正常状态下的文字样式
Misc【其他属性】		
ShowMore	BOOL	当 Item 数量超过可显示数量时，是否出现“更多”按钮
ShowMore【更多按钮属性】		
NormalImage	ImageBase	显示更多的正常图片
SeleCtedImage	ImageBase	显示更多的挑选图片
DisabledImage	ImageBase	显示更多的禁用图片
HotImage	ImageBase	显示更多的高亮图片
Cursor【光标属性】		
Cursor	Cursor	鼠标移动到此控件上显示的光标

4.17 ScrollBar 控件属性

滚动条控件，通常被其他的控件调用，放置在其他控件内部组合使用。如 ListView，SimpleTree。



控件属性名称	属性类型	属性含义
Horz【水平滚动条属性】		
Normal【水平滚动条正常状态属性】		
BackImage	ImageBase	水平的正常状态滚动槽的背景图片
LeftArrow	ImageBase	水平的正常状态滚动向上箭头的图片
RightArrow	ImageBase	水平的正常状态滚动向下箭头的图片
Thumb	ImageBase	水平的正常状态滚动滑块的图片
Tick	ImageBase	水平的正常状态滚动滑块的 Tick 图片
Disable【水平滚动条禁用状态属性】		
BackImage	ImageBase	水平的禁用状态滚动槽的背景图片
LeftArrow	ImageBase	水平的禁用状态滚动向上箭头的图片
RightArrow	ImageBase	水平的禁用状态滚动向下箭头的图片
Thumb	ImageBase	水平的禁用状态滚动滑块的图片
Tick	ImageBase	水平的禁用状态滚动滑块的 Tick 图片
Hot【水平滚动条高亮状态属性】		
BackImage	ImageBase	水平的高亮状态滚动槽的背景图片
LeftArrow	ImageBase	水平的高亮状态滚动向上箭头的图片
RightArrow	ImageBase	水平的高亮状态滚动向下箭头的图片
Thumb	ImageBase	水平的高亮状态滚动滑块的图片
Tick	ImageBase	水平的高亮状态滚动滑块的 Tick 图片
Press【水平滚动条按下状态属性】		
BackImage	ImageBase	水平的按下状态滚动槽的背景图片
LeftArrow	ImageBase	水平的按下状态滚动向上箭头的图片
RightArrow	ImageBase	水平的按下状态滚动向下箭头的图片
Thumb	ImageBase	水平的按下状态滚动滑块的图片

Tick	ImageBase	水平的按下状态滚动滑块的 Tick 图片
Vert【垂直滚动条属性】		
Normal【垂直滚动条正常状态属性】		
BackImage	ImageBase	垂直的正常状态滚动槽的背景图片
LeftArrow	ImageBase	垂直的正常状态滚动向上箭头的图片
RightArrow	ImageBase	垂直的正常状态滚动向下箭头的图片
Thumb	ImageBase	垂直的正常状态滚动滑块的图片
Tick	ImageBase	垂直的正常状态滚动滑块的 Tick 图片
Disable【垂直滚动条禁用状态属性】		
BackImage	ImageBase	垂直的禁用状态滚动槽的背景图片
LeftArrow	ImageBase	垂直的禁用状态滚动向上箭头的图片
RightArrow	ImageBase	垂直的禁用状态滚动向下箭头的图片
Thumb	ImageBase	垂直的禁用状态滚动滑块的图片
Tick	ImageBase	垂直的禁用状态滚动滑块的 Tick 图片
Hot【垂直滚动条高亮状态属性】		
BackImage	ImageBase	垂直的高亮状态滚动槽的背景图片
LeftArrow	ImageBase	垂直的高亮状态滚动向上箭头的图片
RightArrow	ImageBase	垂直的高亮状态滚动向下箭头的图片
Thumb	ImageBase	垂直的高亮状态滚动滑块的图片
Tick	ImageBase	垂直的高亮状态滚动滑块的 Tick 图片
Press【垂直滚动条按下状态属性】		
BackImage	ImageBase	垂直的按下状态滚动槽的背景图片
LeftArrow	ImageBase	垂直的按下状态滚动向上箭头的图片
RightArrow	ImageBase	垂直的按下状态滚动向下箭头的图片
Thumb	ImageBase	垂直的按下状态滚动滑块的图片
Tick	ImageBase	垂直的按下状态滚动滑块的 Tick 图片
Misc【其他属性】		
Enabled	BOOL	是否启动此控件
DesignState	combobox	设计状态，builder 内预览效果
ScrollElapse	Int	滚动速度
ScrollInfo【滚动条信息】		
Min	Int	最小值，默认 0
Max	Int	最大值，默认 100
Positiom	Int	当前位置，用于 Builder 内预览效果
Page	Int	文档高度，用于 Builder 内预览效果

4.18 HeaderCtrl 控件属性

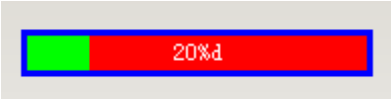
列条控件，通常被其他的控件调用，放置在其他控件内部组合使用。如 ListView，SimpleTree。

控件属性名称	属性类型	属性含义
Back【背景图片】		
DrawColor	BOOL	是否使用颜色来绘制背景，否者使用图片
Color	Colour	背景颜色
Image	ImageBase	背景图片
Item【Header 项属性】		
DrawColor	BOOL	是否使用颜色来绘制 Item 背景，否者使用图片
Normal【正常状态】		
Color	colour	正常状态下的颜色
Image	ImageBase	正常状态下的图片
TextStyle	Text	正常状态下的文字风格
Hot【高亮状态】		
Color	colour	高亮状态下的颜色
Image	ImageBase	高亮状态下的图片
TextStyle	Text	高亮状态下的文字风格
Press【按下状态】		
Color	colour	按下状态下的颜色
Image	ImageBase	按下状态下的图片
TextStyle	Text	按下状态下的文字风格
Disable【禁用状态】		
Color	colour	禁用状态下的颜色
Image	ImageBase	禁用状态下的图片
TextStyle	Text	禁用状态下的文字风格
Misc【其他属性】		
SupportDrag	BOOL	是否支持拖拽来交换 item 的绘制
DividerColor	Color	拖拽条的颜色
SortIcon【排序图标属性】		
ShowSortIcon	BOOL	Item 上是否显示排序图标
AlignText		
ASCArrow	ImageBase	升序图标
DESCArrow	ImageBase	降序图标
Width	Int	图标宽度
Height	Int	图标高度
HorzRight	Int	与右端的间隔
CheckBox【Item 上 CheckBox 控件属性】		
SupportCheckBox	BOOL	是否开启 Item 上的 Checkbox 控件
Offset	Int	控件与左端的间隔
CheckedImage【CheckBox 控件选取属性】		
Normal	ImageBase	选中的正常状态图片
Hot	ImageBase	选中的高亮状态图片
Press	ImageBase	选中的按下状态图片
Disable	ImageBase	选中的禁用状态图片

UncheckedImage【CheckBox 控件未选取属性】		
Normal	ImageBase	未选中的正常状态图片
Hot	ImageBase	未选中的高亮状态图片
Press	ImageBase	未选中的按下状态图片
Disable	ImageBase	未选中的禁用状态图片

4.19 ProgressBar 控件属性

进度条控件



蓝色部分为 Border

红色部分为 BackClient

绿色部分为 Clientground

控件属性名称	属性类型	属性含义
ProgressbarStyle【进度条样式】		
Style	Options	标准样式，Vista 样式，饼图样式
Border【边框样式】		
DrawColorBorder	BOOL	是否画使用颜色绘制背景，否则使用图片
BorderWidth	Int	Client 区域与背景之间的间隔
Image	ImageBase	背景图片
ClrBorder	colour	背景颜色
BackClient【进度背景区域样式】		
DrawColorBack	BOOL	是否使用颜色绘制进度背景区域背景，否则用图片
Image	ImageBase	进度背景区域图片
Color	colour	进度背景区域颜色
Clientground【进度表示区域样式】		
DrawColorBack	BOOL	是否使用颜色绘制进度表示区域背景，否则用图片
Image	ImageBase	进度表示区域图片
Color	colour	进度表示区域颜色
ClientOffset【进度表示区域与进度背景区域间隔】		
Left	Int	左端间隔
Top	Int	上端间隔
Right	Int	右侧间隔
Bottom	Int	下端间隔
Foreground【前景区域设置，Vista 样式有效】		
DrawColorBack	BOOL	是否使用颜色绘制前景，否则使用图片

Image	ImageBase	前景图片
Color	colour	前景颜色
ForeOffset【前景与进度背景区域的间隔】		
Left	Int	左边的间隔
Top	Int	上边的间隔
Right	Int	右边的间隔
Bottom	Int	底边的间隔
ForeWidth【前景的宽度】		
ForeWdith	Int	前景宽度
ClientWdith	Int	滚动区域的宽度
ForeSpeed	Int	前景的滚动速度
ForeElaspe	Int	前景滚动速度
Range【进度条的范围】		
Min	Int	进度条最小值
Max	Int	进度条最大值
Pos	Int	进度当前位置，builder 内预览
Step	Int	进度条每次的前进数量
Type【进度条类型】		
IsHorz	BOOL	是否是水平滚动条
IsRunOnce	BOOL	是否循环滚动一次（Vista 样式有效）
Text【进度条文本】		
TextStyle	TextStly	进度条上显示文本样式
Format	Text	进度条文本格式
DrawText	BOOL	是否显示文本

4.20 SliderBar 控件属性

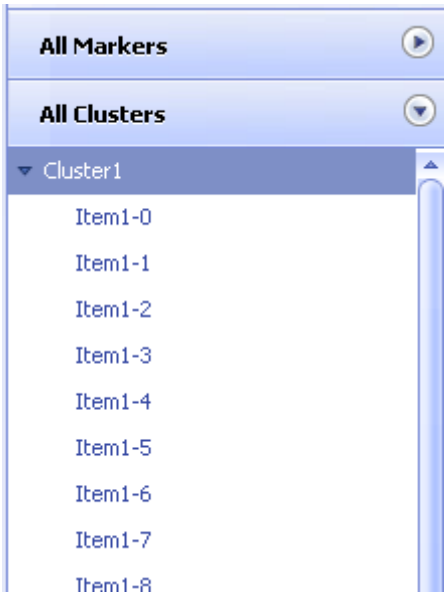
滑条控件

控件属性名称	属性类型	属性含义
SliderType【滑动条类型】		
SliderType	Options	滑动条类型，标准滑动条，自定义滑动条
HorzSlider	BOOL	是否是水平滑动条，否则为垂直
Channel【滑槽属性】		
DrawColor	BOOL	是否使用颜色绘制滑槽
Image	ImageBase	滑槽背景图片
BorderColor	Colour	滑槽边界颜色

BackColor	Colour	滑槽背景颜色
ChannelHeight	Int	滑槽的高度
BorderWidth	Int	滑槽的宽度
MinVal	Int	滑槽最小的数值
MaxVal	Int	滑槽最大的数值
Thumb【滑块属性】		
DrawColor	colour	是否使用颜色绘制滑块
Height	Int	滑块的高度
Width	Int	滑块的宽度
Normal【正常状态下滑块属性】		
Image	imageBase	正常状态下滑块背景图片
Color	colour	正常状态下滑块背景颜色
Press【按下状态下滑块属性】		
Image	imageBase	按下状态下滑块背景图片
Color	colour	按下状态下滑块背景颜色
Disable【禁用状态下滑块属性】		
Image	imageBase	禁用状态下滑块背景图片
Color	colour	禁用状态下滑块背景颜色
Hot【高亮状态下滑块属性】		
Image	imageBase	高亮状态下滑块背景图片
Color	colour	高亮状态下滑块背景颜色

5. OfficeControls 控件属性介绍

5.1. OutLookBar 控件属性

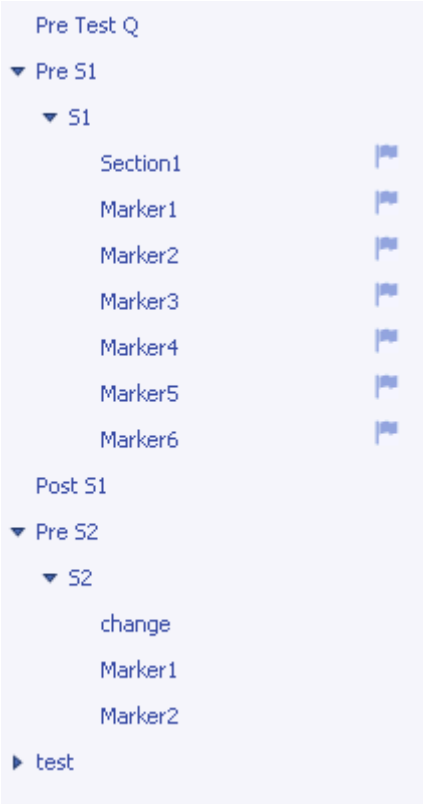


控件属性名称	属性类型	属性含义
Back【背景属性】		
DrawColor	BOOL	是否使用颜色来绘制背景，否则使用图片
BorderSize	Int	背景和 Item 之间的间距
Color	Color	背景颜色
Image	Imagebase	背景图片
Folder【文件夹属性属性】		
DrawColor	BOOL	是否使用颜色来绘制文件夹背景
BorderSize	Int	文件夹背景与前景的间隔
Color	Color	文件夹背景图片
Normal【文件夹正常状态属性】		
BorderColor	Color	文件夹正常状态下背景颜色
BackColor	Color	文件夹正常状态下前景颜色
Image	ImageBase	文件夹正常状态下背景图片
Graphics	ImageBase	文件夹正常状态下左侧图标
GraphicsExpand	ImageBase	文件夹正常状态下展开后左侧图标
TextStyle	TextStyle	文件夹正常状态下文本样式
Hot【文件夹高亮状态属性】		
BorderColor	Color	文件夹高亮状态下背景颜色
BackColor	Color	文件夹高亮状态下前景颜色
Image	ImageBase	文件夹高亮状态下背景图片
Graphics	ImageBase	文件夹高亮状态下左侧图标
GraphicsExpand	ImageBase	文件夹高亮状态下展开后左侧图标
TextStyle	TextStyle	文件夹高亮状态下文本样式
Press【文件夹按下状态属性】		
BorderColor	Color	文件夹按下状态下背景颜色
BackColor	Color	文件夹按下状态下前景颜色
Image	ImageBase	文件夹按下状态下背景图片
Graphics	ImageBase	文件夹按下状态下左侧图标
GraphicsExpand	ImageBase	文件夹按下状态下展开后左侧图标
TextStyle	TextStyle	文件夹按下状态下文本样式
Disable【文件夹禁用状态属性】		
BorderColor	Color	文件夹禁用状态下背景颜色
BackColor	Color	文件夹禁用状态下前景颜色
Image	ImageBase	文件夹禁用状态下背景图片
Graphics	ImageBase	文件夹禁用状态下左侧图标
GraphicsExpand	ImageBase	文件夹禁用状态下展开后左侧图标
TextStyle	TextStyle	文件夹禁用状态下文本样式
Graphcis【文件夹图标属性】		
UpDownMode	BOOL	图标是否垂直居中否则水平居中
RightMode	BOOL	图标是否靠右，否则靠左
X	Int	图标水平偏移量
Y	Int	图标垂直偏移量
Item【文件夹 Item】		

DrawColor	BOOL	是否使用颜色绘制 Item 背景
ItemHeight	Int	Item 的高度
ItemSpace	Int	Item 之间的间隔
Normal【Item 正常状态属性】		
Image	ImageBase	Item 正常状态背景图片
Color	Color	Item 正常状态背景颜色
TextStyle	TextStyle	Item 正常状态的文本样式
Hot【Item 高亮状态属性】		
Image	ImageBase	Item 高亮状态背景图片
Color	Color	Item 高亮状态背景颜色
TextStyle	TextStyle	Item 高亮状态的文本样式
Press【Item 按下状态属性】		
Image	ImageBase	Item 按下状态背景图片
Color	Color	Item 按下状态背景颜色
TextStyle	TextStyle	Item 按下状态的文本样式
Disbale【Item 禁用状态属性】		
Image	ImageBase	Item 禁用状态背景图片
Color	Color	Item 禁用状态背景颜色
TextStyle	TextStyle	Item 禁用状态的文本样式
Arrow【Item 更多箭头属性，Item 数量大于可显示数量出现箭头】		
ArrowSpace	Int	上下箭头之间的间隔
Normal【箭头正常状态属性】		
UpArrowImage	ImageBase	上箭头背景
DownArrowImage	ImageBase	下箭头背景
Hot【箭头高亮状态属性】		
UpArrowImage	ImageBase	上箭头背景
DownArrowImage	ImageBase	下箭头背景
Press【箭头按下状态属性】		
UpArrowImage	ImageBase	上箭头背景
DownArrowImage	ImageBase	下箭头背景
Disable【箭头禁用状态属性】		
UpArrowImage	ImageBase	上箭头背景
DownArrowImage	ImageBase	下箭头背景

5.2. SimleTree 控件属性

树形控件



控件属性名称	属性类型	属性含义
Back【背景属性】		
DrawColor	BOOL	是否使用颜色来绘制背景，否则使用图片
Image	ImageBase	背景图片
Color	Color	背景颜色
Item【树节点属性】		
DrawColor	BOOL	是否使用颜色来绘制 Item 背景，否则使用图片
Height	Int	Item 的高度
Normal【Item 正常状态下属性】		
Image	ImageBase	正常状态下 Item 背景图片
Color	Color	正常状态下 Item 背景颜色
TextStyle	TextStyle	正常状态下 Item 文本样式
Hot【Item 高亮状态下属性】		
Image	ImageBase	高亮状态下 Item 背景图片
Color	Color	高亮状态下 Item 背景颜色
TextStyle	TextStyle	高亮状态下 Item 文本样式
Press【Item 按下状态下属性】		
Image	ImageBase	按下状态下 Item 背景图片
Color	Color	按下状态下 Item 背景颜色
TextStyle	TextStyle	按下状态下 Item 文本样式
Disable【Item 禁用状态下属性】		
Image	ImageBase	禁用状态下 Item 背景图片
Color	Color	禁用状态下 Item 背景颜色
TextStyle	TextStyle	禁用状态下 Item 文本样式
Uncheked【未展开 Item 的图标属性】		

Normal【正常状态下】		
ButtonImage	ImageBase	Item 正常状态下未展开图标
CheckImage	ImageBase	未选中 Checkbox 正常状态图标
Hot【高亮状态下】		
ButtonImage	ImageBase	Item 高亮状态下未展开图标
CheckImage	ImageBase	未选中 Checkbox 高亮状态图标
Press【按下状态下】		
ButtonImage	ImageBase	Item 按下状态下未展开图标
CheckImage	ImageBase	未选中 Checkbox 按下状态图标
Disable【禁用状态下】		
ButtonImage	ImageBase	Item 禁用状态下未展开图标
CheckImage	ImageBase	未选中 Checkbox 禁用状态图标
Checked【展开 Item 的图标属性】		
Normal【正常状态下】		
ButtonImage	ImageBase	Item 正常状态下展开图标
CheckImage	ImageBase	选中 Checkbox 正常状态图标
Hot【高亮状态下】		
ButtonImage	ImageBase	Item 高亮状态下展开图标
CheckImage	ImageBase	选中 Checkbox 高亮状态图标
Press【按下状态下】		
ButtonImage	ImageBase	Item 按下状态下展开图标
CheckImage	ImageBase	选中 Checkbox 按下状态图标
Disable【禁用状态下】		
ButtonImage	ImageBase	Item 禁用状态下展开图标
CheckImage	ImageBase	选中 Checkbox 禁用状态图标
Misc【其他属性】		
DragSupport	BOOL	是否开启拖拽改变 Item 的顺序
HasButtons	BOOL	是否绘制展开和收缩图标
RootHasButtons		如果节点没有子节点是否仍然绘制展开和收缩图标
HasLines	BOOL	界面之间是否绘制连线
LineOnRoot	BOOL	是否绘制根节点之间的连线
LineStyle	Options	连线的类型
LineColor	Color	连线的颜色
LineSize	Int	连线的宽度
Indent	Int	两个节点之间的间距
HasCheckStyle	BOOL	节点前面是否有 ChekcBox
SupportMultiSel	BOOL	是否支持多选
Graphics【图标属性】		
ShowIcon	BOOL	节点是否显示图标
ShowCenter	BOOL	是否将图标自动调整到节点中央
AlignToRIght	BOOL	图标是否放置在节点文本的右侧
OffsetX	Int	图标的水平偏移量
OffsetY	Int	图标的垂直偏移量
StateIcon【状态图标属性，已过时】		
ShowState	BOOL	无效

OffsetX	Int	无效
OffsetY	Int	无效
NormalImage	ImageBase	无效
HotImage	ImageBase	无效
PressImage	ImageBase	无效
DisableImage	ImageBase	无效
ScrollBarInfo【滚动条信息】		
ScrollSize	Int	滚动条宽度
LeftScroll	BOOL	滚动条是否处于控件的左侧
Header【HeaderCtrl 信息】		
HaveHead	BOOL	是否拥有 Header
HeadCtrl	Text	HeaderCtrl 的名称
ToolBar【节点右侧有拥有工具栏】		
HasToolBar	BOOL	是否启用工具栏
Offset	Int	工具栏的水平偏移量
MinSpaceWidhtText	Int	工具栏与文本之间的间距
ToolBar【节点右侧有拥有工具栏】		
DragCursor	Cursor	拖拽 Item 时的光标
Cursor	Cursor	鼠标移动到此控件上的光标

5.3. TaskPanel 控件属性

控件属性名称	属性类型	属性含义
Background【背景属性】		

DrawBackColor	BOOL	是否使用颜色来绘制背景，否则使用图片
NormalColor	Color	启动时背景颜色
NormalImage	ImageBase	启用时背景图片
DisableColor	Color	禁用是背景颜色
DisableImage	ImageBase	禁用是背景图片
LefeMargin	Int	Item 与控件的左间距
TopMargin	Int	Item 与控件的上间距
RightMargin	Int	Item 与控件的右间距
BottomMargin	Int	Item 与控件的下间距
Group【组属性】		
GroupSpace	Int	组与组之间的间距
CaptionHeight	Int	组的高度
IsDrawClintColor	BOOL	是否使用颜色来绘制组的背景区域
ClientAreaImage	ImageBase	组背景区域的图片
ClientAreaColor	Color	组背景区域的颜色
LeftMargin	Int	无效
TopMargin	Int	无效
RightMargin	Int	无效
BottomMargin	Int	无效
Icon【组图标属性】		
AlignRight	BOOL	图标是否放置在组文本的右侧
IconOffsetX	Int	图标的水平偏移量
ExpandIconImage	ImageBase	展开图标
UnexpandIconImage	ImageBase	收缩图标
CaptionIconOffsetX	Int	图标的水平偏移量
CaptionIconOffsetY	Int	图标的垂直偏移量
CaptionImages【组背景属性】		
Normal	ImageBase	正常状态组背景图片
Disable	ImageBase	禁用状态组背景图片
Hot	ImageBase	高亮状态组背景图片
Press	ImageBase	按下状态组背景图片
Focus	ImageBase	选择状态组背景图片
CaptionImages【组背景属性】		
Normal	TextStyle	正常状态组文字样式
Disable	TextStyle	禁用状态组文字样式
Hot	TextStyle	高亮状态组文字样式
Press	TextStyle	按下状态组文字样式
Focus	TextStyle	选择状态组文字样式
Item【组 Item 属性】		
ItemSpace	Int	Item 之间的间距
ItemHegiht	Int	Item 的高度
IconOffsetX	Int	图标的水平偏移量
ItemImages【Item 的背景】		
Normal	ImageBase	Item 的正常状态背景图片
Disable	ImageBase	Item 的禁用状态背景图片

Hot	ImageBase	Item 的高亮状态背景图片
Press	ImageBase	Item 的按下状态背景图片
Focus	ImageBase	Item 的选择状态背景图片
ItemTextStyles【Item 的文字样式】		
Normal	TextStyle	Item 的正常状态文字样式
Disable	TextStyle	Item 的禁用状态文字样式
Hot	TextStyle	Item 的高亮状态文字样式
Press	TextStyle	Item 的按下状态文字样式
Focus	TextStyle	Item 的选择状态文字样式

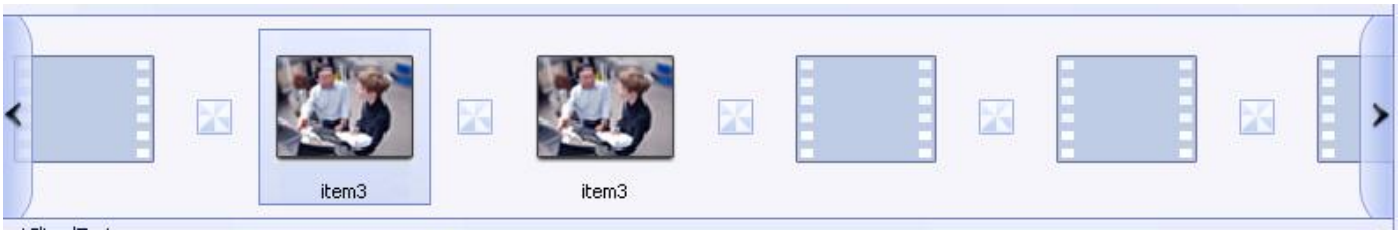
6. AdvancedControls 控件属性介绍

6.1. Subtitle 控件属性

此控件已过时，使用自定义模式的 Listview 可代替此控件

6.2. ScrollFairy 控件属性

滚动图片控件



控件属性名称	属性类型	属性含义
Back【背景属性】		
DrawColor	BOOL	是否使用颜色来绘制背景，否则使用图片
BackImage	ImageBase	背景图片
BackColor	Color	背景颜色
Item【Item 属性】		
AutoSize	BOOL	
ShowText	BOOL	是否在 Item 下显示文本
TopOffset	Int	Item 垂直方向的偏移量
ItemWidht	Int	Item 的宽度
ItemHeight	Int	Item 的高度
ItemSpace	Int	Item 之间的间距
Normal【Item 正常状态属性】		
BackImage	ImageBase	Item 正常状态背景图片

FrontImage	ImageBase	Item 正常状态前景图片
TextImage	TextStyle	Item 正常状态文字样式
Hot【Item 高亮状态属性】		
BackImage	ImageBase	Item 高亮状态背景图片
FrontImage	ImageBase	Item 高亮状态前景图片
TextImage	TextStyle	Item 高亮状态文字样式
Press【Item 按下状态属性】		
BackImage	ImageBase	Item 按下状态背景图片
FrontImage	ImageBase	Item 按下状态前景图片
TextImage	TextStyle	Item 按下状态文字样式
Disable【Item 禁用状态属性】		
BackImage	ImageBase	Item 禁用状态背景图片
FrontImage	ImageBase	Item 禁用状态前景图片
TextImage	TextStyle	Item 禁用状态文字样式
InfoPanel【信息面板属性】		
SupportExpand	BOOL	信息面板是否具有展开功能
Direction	Options	展开方向，上下左右
InfoPanelModel	Dialog	信息面板模板，弹出窗口选择一个 UIForm
ShrinkSize	Int	缩进的大小
ExpandSize	Int	展开的大小
ShrinkSpeed	Int	缩进的速度
ExpandSpeed	Int	展开的速度
CtrlPanel【控制面板属性】		
AutoHide	BOOL	是否自动隐藏控制面板
ShowSpeed	Int	控制面板显示速速（开启自动隐藏后有效）
HideSpeed	Int	控制面板隐藏速速（开启自动隐藏后有效）
LeftPanel	Text	左面板名称
LeftCtrlButton	Text	左控制按钮名称
RightPanel	Text	右面板名称
RightCtrlButton	Text	右控制按钮名称
Indicator【】		
BShow	BOOL	是否显示指示器
IndicatorBack	ImageBase	指示器背景图片
IndicatorDark	ImageBase	是否显示指示器
IndicatorLight	ImageBase	是否显示指示器
SpaceItem【间隔项】		
HasSpeaceItem	BOOL	是否拥有间隔项
DefaultBackground	ImageBase	间隔项的默认背景
Widht	Int	间隔项的宽度
Height	Int	间隔项的高度
Misc【其他属性】		
Horz	BOOL	是否是水平的
SupportDrag	BOOL	是否可拖拽改变 Item 的位置
MultiPages	BOOL	
Cycle	BOOL	是否可循环显示

AutoScroll	BOOL	是否自动滚动
AutoStop	BOOL	是否自动停止
AutoScrollSpeed	Int	自动滚动速度，毫秒
ScrollSpeed	Int	滚动速度，毫秒
IconAutoSize	BOOL	自动调整 Icon 的大小
CanSelectedItem	BOOL	是否可选择 Item

6.3. ScrollChannel 控件属性

此控件已过时，使用自定义模式的 Listview 可代替此控件

7. IndustryControls 控件属性介绍

7.1. Knob 控件属性

旋钮控件

控件属性名称	属性类型	属性含义
Knob【背景属性】		
Cycle	BOOL	是否可不断的旋转旋钮
MinVal	Int	表示的最小值
MaxVal	Int	表示的最大值
Normal【正常状态】		
Back	ImageBase	背景图片
Fore	ImageBase	旋钮图片
Hot【高亮状态】		
Back	ImageBase	背景图片
Fore	ImageBase	旋钮图片
Press【按钮状态】		
Back	ImageBase	背景图片
Fore	ImageBase	旋钮图片
Disable【禁用状态】		
Back	ImageBase	背景图片
Fore	ImageBase	旋钮图片

7.2. LEDCtrl 控件属性

使用图片来代替文字

控件属性名称	属性类型	属性含义
Back	ImageBase	背景图片
Fore	ImageBase	前景图片
Text	Text	表示的文本
ImageChars	Text	前景图片对应的文本

ImageChars	Text	前景图片对应的文本

7.3. IndicatorCtrl 控件属性

指示灯控件

控件属性名称	属性类型	属性含义
IndicatorCtrl【背景属性】		
Blink	BOOL	是否自动闪烁
BlinkSpace	Int	闪烁的速度
Normal【闪烁图片一】		
Back	ImageBase	闪烁图片一
Yellow【闪烁图片二】		
Back	ImageBase	闪烁图片二
Green【闪烁图片三】		
Back	ImageBase	闪烁图片三
Yellow【闪烁图片四】		
Back	ImageBase	闪烁图片四
Icon【闪烁图片属性】		
AutoSize	BOOL	自动调整大小
AutoWidht	Int	自动调整的宽度
AutoHeight	Int	自动调整的高度
AutoCenter	BOOL	是否居中显示
OffsetX	Int	水平偏移量
OffsetY	Int	垂直偏移量
AlignTop	BOOL	是否靠于顶端

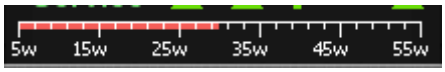
7.4. DialCtrl 控件属性

仪表盘控件

控件属性名称	属性类型	属性含义
Dial【背景属性】		
DrawColor	BOOL	是否使用颜色或者图片来绘制背景
DialImage	ImageBase	表盘背景图片
ArtLineWidht	Int	刻度线宽度
Positive	Color	刻度线正值的颜色，通过临界值判断
Negative	Color	刻度线负值的颜色，通过临界值判断
Pointer【指针属性】		
Pointer	ImageBase	指针的图片
FastX	Int	水平偏移量
FastY	Int	垂直偏移量
Origin	ImageBase	远点图片
Tics【刻度线属性】		
TextStyle	TextStyle	刻度线文本样式
ShowHalfTag	BOOL	
BigFrequery	Int	大分割间距
BigTicLength	Int	大分割线长度
BigTicWidht	Int	大分割线宽度
SmallFrequery	Int	小分割间距
SmallTicLength	Int	小分割线长度
SmallTicWidht	Int	小分割线宽度
Misc【其他属性】		
Radius	Int	刻度盘半径
MaxValue	Int	最大值
MinValue	Int	最小值
Rate	Int	
ArcAngle		刻度盘的角度
CriticalValue	Int	临界值位置

7.5. Thermometer 控件属性

温度计控件



控件属性名称	属性类型	属性含义
Knob【背景属性】		
bLIN	BOOL	是否可不断的旋转旋钮
MinVal	Int	表示的最小值

六、 DirectUI 界面库调用方法

1. 如何调用控件的方法

1.1 皮肤界面资源类型

DirectUI 界面库中有 10 种类型的资源，包括：

资源类型	资源定义
图片资源	DUIOBJTYPE_IMAGE
ImageBase 资源	DUIOBJTYPE_IMAGEBASE
文字样式资源	DUIOBJTYPE_TEXTSTYLE
字体资源	DUIOBJTYPE_FONTEX
光标资源	DUIOBJTYPE_CURSOR
DirectUI 窗口资源	DUIOBJTYPE_DIRECTUI
渐变颜色资源	DUIOBJTYPE_GRADIENTCLR
图片列表资源（已过时）	DUIOBJTYPE_IMAGELIST
图像区域资源	DUIOBJTYPE_SKINRGN
控件资源	DUIOBJTYPE_PLUGIN

1.2 创建 DirectUI 窗口

通过 DUIRes 对象的 CreateDirectUI 接口我们可以将 DirectUI 绑定到一个窗口，并返回一个 IDirectUI 对象指针。

IDirectUI* m_pDirectUI = m_pDUIRes->CreateDirectUI(_T("main"),hWnd);

IDirectUI* CreateDirectUI（LPCTSTR strDirectUIName, HWND hWnd）

strDirectUIName：Builder 内设置 DirectUI 窗口的名称

hWnd：需要绑定窗口的窗口句柄

1.3 获取皮肤资源

通过 DUIRes 对象的 GetResObject 接口我们就可以通过获得不同类型的资源。

如获得 Imagebase 资源：

IDUIImageBase* pImageBase =

（IDUIImageBase*）m_pDUIRes->GetResObject(DUIOBJTYPE_IMAGEBASE, _T("ImageBaseI"), NULL, TRUE);

IDUIObj* GetResObject（DUIObjType eType, LPCTSTR strName, ISkinObjResBase * pParent, BOOL bDeep）

eType：资源类型

strName：资源的名称

pParent：父控件，取控件资源时有效

bDeep：是否递归查找，取控件资源时有效

1.4 获取控件对象

有两种取控件对象的方法，第一种通过资源管理对象取控件资源来提取。

ICmdButton* pButton =

（ICmdButton *）m_pDUIRes->GetResObject(DUIOBJTYPE_PLUGIN, _T("Button1"), NULL, TRUE);

第二种通过控件的父对象

ICmdButton* pButton =

（ICmdButton *）m_pDirectUI->GetObjectByCaption(DUIOBJTYPE_PLUGIN, _T("Button1"), TRUE);

ISkinObjResBase* GetResObject（DUIObjType eType, LPCTSTR strName, BOOL bDeep）

eType：资源类型

strName：控件的名称

bDeep：是否递归查找

1.5 调用控件对象接口

成功获取控件对象指针后可直接调用控件接口。

pButton->SetText(_T("PushButton"));

2. 如何响应控件的事件

通过接受自定义消息我们可处理控件发送给逻辑程序的事件

2.1 平台消息

消息名称	DUISM_LBUTONUP
消息描述	平台左键释放消息
WPARAM	IDUIContorlBase* 点击控件的指针
LPARAM	INT 资源的 id

消息名称	DUISM_RBUTONDOWN
消息描述	平台右键按下消息
WPARAM	IDUIContorlBase* 点击控件的指针
LPARAM	INT 资源的 id

消息名称	DUISM_LBUTONDOWN
消息描述	平台左键按下消息
WPARAM	IDUIContorlBase* 点击控件的指针
LPARAM	INT 资源的 id

消息名称	DUISM_LBUTTONDBCLICK
消息描述	平台左键双击消息
WPARAM	IDUIContorlBase* 点击控件的指针
LPARAM	INT 资源的 id

消息名称	DUISM_CHANGESKINSTART
消息描述	开始换肤消息
WPARAM	-
LPARAM	-

消息名称	DUISM_CHANGESKIN
消息描述	换肤结束消息
WPARAM	-
LPARAM	-

2.2 Combox 自定义消息

消息名称	DUI_CBN_CLICKBUTTON
消息描述	点击下拉按钮消息
WPARAM	控件指针
LPARAM	-

消息名称	DUI_CBN_CLICKEDIT
消息描述	点击文本区域消息
WPARAM	控件指针
LPARAM	-

消息名称	DUI_CBN_SELCHANGE
消息描述	选择改变消息
WPARAM	控件指针
LPARAM	MAKELONG(nIndexNew, nIndexOld)

消息名称	DUI_CBN_DROPDOWN
消息描述	下拉框下拉消息
WPARAM	控件指针
LPARAM	-

消息名称	DUI_CBN_CLOSEUP
消息描述	下拉框收起消息
WPARAM	控件指针
LPARAM	-

2.3 FormBorder 控件自定义消息

消息名称	DUIFB_CAPTIONDBCLICK
消息描述	双击标题栏消息
WPARAM	-

LPARAM	-

2.4 ListView 控件自定义消息

消息名称	DUILV_NOTIFY
消息描述	Listview 自定义消息
WPARAM	DUILVNotifyInfo*
LPARAM	-

```
DUILVNotifyInfo
    {
    ListViewMsgID eDUILVMsgId;
    LONG lParam1;
    LONG lParam2;
    LONG lParam3;
    LONG lParam4;
    LONG lParam5;
}    DUILVNotifyInfo;
```

eDUILVMsgId	DUI_LV_HEADER_LBUTTONUP
消息描述	点击 report 样式下的 header 列
lparam1	listivew 控件指针
lparam2	列序号
lparam3	-
lparam4	-
lparam5	-

eDUILVMsgId	DUI_LV_LBUTTONDOWN
消息描述	左键按下
lparam1	Listview 对象指针
lparam2	Item 对象指针
lparam3	控件的 id
lparam4	Item 的 id
lparam5	-

eDUILVMsgId	DUI_LV_RBUTTONDOWN
消息描述	右键按下
lparam1	Listview 对象指针
lparam2	Item 对象指针
lparam3	控件的 id
lparam4	Item 的 id
lparam5	-

eDUILVMsgId	DUI_LV_RBUTTONUP
消息描述	右键弹起
lparam1	Listview 对象指针
lparam2	Item 对象指针
lparam3	控件的 id
lparam4	Item 的 id
lparam5	-

eDUILVMsgId	DUI_LV_CUSTOMITEM_CLICK
消息描述	右键弹起
lparam1	Listview 对象指针
lparam2	模板控件指针
lparam3	组的 id
lparam4	Item 的 id
lparam5	模板控件的名称

2.5 Tab 控件自定义消息

消息名称	DUI_TABMSG_SELCHANGED
消息描述	Item 改变消息
WPARAM	Item 的序号
LPARAM	控件对象指针

消息名称	DUI_TABMSG_SELIDCHANGED
消息描述	Item 改变消息
WPARAM	Item 的序号
LPARAM	控件对象指针

消息名称	DUI_TABMSG_CLOSE
消息描述	Item 关闭消息
WPARAM	Item 的序号
LPARAM	控件对象指针

2.6 Sliderbar 控件自定义消息

消息名称	DUISM_SLIDER_THUMBTRACK
消息描述	Thumb 位置改变消息
WPARAM	控件对象指针
LPARAM	当前值

2.7 Toolbar 控件自定义消息

消息名称	DUITB_ITEM_LBUTTONDOWN
消息描述	Toolbar item 的按下消息
WPARAM	Item 的 id
LPARAM	toolbar 控件指针

消息名称	DUITB_ITEM_LBUTTONUP
消息描述	Toolbar item 的按下释放消息
WPARAM	Item 的 id
LPARAM	toolbar 控件指针

消息名称	DUITB_ITEM_COMMAND
消息描述	Toolbar item 点击消息
WPARAM	Item 的 id
LPARAM	toolbar 控件指针

3. DirectUI 界面库控件结构

ICmdButton → IDUIControlBase → ISkinObjResBase → IDUIObj
IDUIAnimate →
IDUICheckBox →
IUIForm →
IRadioBox →
IDUIHwnObj →
IDUIListView →
IDUIMenuBar →
IDUIProgressbar →
IDUIScrollbar →
IDUIStatic →
IDUITabCtrl →
IDUIToolBar →
IDUIStarCtrl →
IDUIPopupMenu →
IDUIComboBox →
IDUILogoObj →
IDUIFormBorder →
IDUICanlendar →
IDUISimpleTree →
IDUIOutlookBar →

4. 平台类接口

4.1 IDUIObj

IDUIObj 是 DirectUI 中所有对象的基类

从 IDUIObj 派生的对象有 ISkinObjResBase

HRESULT GetName（BSTR* pstrResult）
获取对象名称
pstrResult 返回对象的名称

HRESULT SetName（BSTR strResult）
设置对象名称
strResult 对象的名称

HRESULT SetObjPtr（LONG pObjPtr）
设置对象中的 C++对象
pObjPtr C++对象指针

HRESULT GetObjPtr（LONG* pObjPtr）
设置对象中的 C++对象
pObjPtr 返回 C++对象指针

HRESULT GetObjectId（BSTR* pStrGUID）
获取对象的 GUID 值
pStrGUID 返回对象的 GUID 字符串

HRESULT AccessConfig([in]VARIANT_BOOL bRead, [out,retval]VARIANT_BOOL* pbResult)

对象的进行序列化
bRead TRUE 将对象的属性写入 XML FALSE 从 XML 中读取对象的属性 pbResult 返回成功结果

HRESULT GetDUIRes([out,retval]IDUIRes **ppResult)
获取资源管理对象指针
ppResult 返回的资源管理对象

HRESULT SetDUIRes ([out,retval]IDUIRes *pResult)
设置资源管理对象指针
pResult 设置的资源管理对象指针

HRESULT GetXMLNode([out,retval]OLE_HANDLE*ppXmlEle)
获取对象对应的 XML 节点
ppXmlEle 返回对象对应的 XML 节点

HRESULT GetType([out,retval]DUIObjType* pType)
返回对象类型
pType 返回对象类型

HRESULT SetType([in]DUIObjType objType)
设置对象的类型
objType 设置对象类型

HRESULT GetTypeName([out,retval]BSTR* pstrResult)
获取对象的类型
pstrResult 获取对象类型名称

HRESULT SetTypeName([out,retval]BSTR strTypeName)
设置对象的类型
strTypeName 对象类型名称

HRESULT SetModified([in]VARIANT_BOOL bModified)
设置对象是否被修改，Builder 内调用
bModified VARIANT_TRUE 已被修改 VARIANT_FALSE 未被修改

HRESULT IsModified([out,retval]VARIANT_BOOL *pbModified)
返回对象是否被修改
pbModified VARIANT_TRUE 已被修改 VARIANT_FALSE 未被修改

4.2 ISkinObjResBase ： IDUIObj

从 IskinObjResBase 派生的对象有 IDUIControlBase

ISkinObjResBase 是所有控件的基类

HRESULT IsEnable([out,retval] VARIANT_BOOL* pbResult)
控件是否启用
pbResult 返回启用结果是否启用

HRESULT EnableObject([in] VARIANT_BOOL bEnable,[in]VARIANT_BOOL bRedraw,[out,retval] VARIANT_BOOL* pbResult)
启用或禁用控件
bEnable 启用或禁用控件 bRedraw 是否立即刷新控件

HRESULT SetVisible([in] VARIANT_BOOL bVisible, [in] VARIANT_BOOL bRefresh, [in] VARIANT_BOOL bAnimate)

设置控件是否可见
BVisiable 控件是否可见 bRefresh 是否立即刷新（无效参数，直接设置为 VARIANT_FALSE） bAnimate 是否出现动画（无效参数，直接设置为 VARIANT_FALSE）

HRESULT GetParent([out,retval] ISkinObjResBase** pObjParent)
获取控件的父控件
pObjParent 返回控件的父控件

HRESULT SetParent([in] ISkinObjResBase* pObjParent)
设置控件的父控件
pPobParent 父控件指针

HRESULT GetRoot([out,retval] ISkinObjResBase** pObjRoot)
获取最顶级的控件，一般为 DirectUI 控件
pObjRoot 返回控件的指针

HRESULT IsExisted([in] ISkinObjResBase* pObj, [out,retval] VARIANT_BOOL* pbResult)
检查是否拥有某个子控件，此方法不会递归查找，仅仅查找第一级
pObj 需要查询的控件 pbResult 返回查询结果

HRESULT AddSubObject([in] ISkinObjResBase* pSubObj, [in] VARIANT_BOOL bInsertFirst, [out,retval] VARIANT_BOOL* pbResult)
向控件内添加一子控件

pSubObj 需要添加的子控件指针 bInsertFirst 是否插入为第一项 pbResult 返回插入结果

HRESULT MoveAhead([in] ISkinObjResBase* pSubObj, [out,retval] VARIANT_BOOL* pbResult)
将子控件项前移，控件的前后关系确定了绘制顺序，靠后的控件会后绘制，可能会盖住前面的子控件。
pSubObj 需要调整的子控件

HRESULT MoveBack([in] ISkinObjResBase* pSubObj, [out,retval] VARIANT_BOOL* pbResult)
将子控件项后移，控件的前后关系确定了绘制顺序，靠后的控件会后绘制，可能会盖住前面的子控件。
pSubObj 需要调整的子控件 pbResult 返回调整的结果

HRESULT Remove([out,retval] VARIANT_BOOL* pbResult)
将自己从控件表中删除

HRESULT GetSubObject([in] short nIndex, [out,retval] ISkinObjResBase** pSubObj)
通过索引获取子控件
nIndex：索引序号 pSubObj：返回的子控件指针

HRESULT RemoveAllSubObjects(void)
删除所有的子控件

HRESULT GetSubObjIndex([in] ISkinObjResBase* pSubObj, [out,retval] short* pnIndex)

获取子控件的序号
pSubObject 子控件的指针 pnIndex 返回子控件的序号

HRESULT GetChildCount([out,retval] SHORT* pnCount)
获取子控件的数量
PnCount 返回子控件的数量

HRESULT GetSubCaptionObject([in] BSTR strCaption, [in] SHORT* pnIndex, [out,retval] ISkinObjResBase** pSubObj)
通过子控件的名称和序号来获取子控件和子控件序号
StrCaption 子控件的名称 PnIndex 返回的子控件序号 pSubObj 返回子控件指针

HRESULT GetSubResObject([in] ISkinObjResBase* pObj,[in] short* pnIndex, [out,retval] ISkinObjResBase** pSubObj)
获取子控件的序号
pObj 需要查询的子控件 pnIndex 返回子控件的序号 pSubObj 返回查询到的子控件

HRESULT GetObjectByCaption([in] long lType, [in] BSTR strName, [in] VARIANT_BOOL bDeep, [out,retval] ISkinObjResBase** pObj)
通过控件的名称来获取子控件指针
lType 控件内省，必须为 DUIOBJTYPE _PLUGIN strName 控件名称 bDeep 是否递归查找 pObj 返回查询到的子控件

HRESULT DelAllSubObjects(void)
删除所有的字控件

HRESULT SetPosition([in] IDUIPos* pObjPos)
设置控件的布局

HRESULT ResizeThis(void)
强制刷新一次控件布局，向控件发送 DUI_SIZE 消息

HRESULT RemoveSubObject([in] ISkinObjResBase* pSubObj, [out,retval] VARIANT_BOOL* pbResult)
删除某一个子控件
pSubObj 需要删除的子控件指针 pbResult 返回删除的结果

HRESULT InsertSubObject([in] ISkinObjResBase* pSubObj, [in] short nIndex, [out,retval] short* pnSubIndex)
插入子控件
pSubObj 需要插入的子控件指针 nIndex 插入的位置，从 0 开始 pnSubIndex 返回的插入的位置

HRESULT RedrawWindow([in] VARIANT_BOOL bUpdateNow, [out,retval] VARIANT_BOOL* pbResult)
重绘控件所在的窗口

HRESULT DirectDraw([in] VARIANT_BOOL bDestroyBmp)

重绘控件
bDestroyBmp 是否删除控件的位图缓存

HRESULT GetDirectUI([out,retval] IDispatch** pDirectUI)
获取控件所在的 DirectUI 对象指针
pDirectUI 返回 DirectUI 对象指针

HRESULT GetWindow([out,retval] LONG* pHwnd)
获取控件所在窗口的句柄
pHwnd 返回窗口句柄

HRESULT SetOwnerHwnd([in] LONG lhWnd)
设置控件及其所有的子控件的窗口句柄
lhWnd 设置窗口句柄

HRESULT SetDirectUIHwnd([in] LONG lhWnd)
设置 DirectUI 窗口句柄

HRESULT GetRect([out,retval] IDUIRect** pRect)
获取控件在窗口上的范围
pRect 返回控件区域对象

HRESULT GetDockType([out,retval] DUIDockObjType* pnType)

获取控件的停靠类型
PnType 返回停靠类型

HRESULT Dock([in] DUIDockObjType ntype, [in] VARIANT_BOOL bFull)
设置控件的停靠类型
nType 停靠类型

HRESULT GetPosition([out,retval] IDUIPos** pObjPos)
获取控件的位置

HRESULT IsLButtonDowned([out,retval]VARIANT_BOOL *pbResult)
左键是否点击在此控件上
pbResult 返回结果

HRESULT SetLButtonDowned([in]VARIANT_BOOL bValue)
设置控件是否被点击

HRESULT IsMouseMove([out,retval]VARIANT_BOOL *pbResult)
鼠标是否移入到此控件内
PbResult 返回查询结果

HRESULT SetMouseMove([in]VARIANT_BOOL bValue)

设置鼠标的移入变量

HRESULT IsVisible([out,retval]VARIANT_BOOL *pbResult)
控件是否可见
PbResult 查询结果

HRESULT SetOpacity([in]SHORT nOpacity)
设置控件的透明度
nOpacity 透明度 0~255

HRESULT GetOpacity([out,retval]SHORT *pResult)
获取控件的透明度
pResult 返回透明度

HRESULT SetBlendWithParent([in]VARIANT_BOOL bBlend)
设置控件是否和父控件进行混合
bBlend 是否开启

HRESULT IsBlendWithParent([out,retval]VARIANT_BOOL *pbResult)
查询控件是否开启与父控件进行混合
pbResult 返回结果

HRESULT GetLayerBack([in]SkinRect rect,[out,retval]ILayerGraphics **pResult)
获取控件的缓存区域

Rect 需要获取的区域 ILayerGraphics 返回缓存对象指针

HRESULT GetLayerFore([in]SkinRect rect,[out,retval]ILayerGraphics **pResult)

HRESULT GetLayerMe([out,retval]ILayerGraphics **pResult)
获取缓存对象指针
pResult 返回缓存对象

HRESULT IsSizing([out,retval]VARIANT_BOOL *pbResult)
查询控件是否在调整大小过程中

HRESULT GetDirectUILayerCache([out,retval]ILayerGraphics **pResult)
获取控件所在 DirectuI 对象缓存图层对象

HRESULT UpdateDirectUILayerCache([in]SkinRect rcDest,[in]OLE_HANDLE hDC,[in]SkinRect rcSource,[out,retval]VARIANT_BOOL *pbResult)
更新 DirectUI 对象缓存区域
rcDest 目标区域 hDC 更新 HDC SkinRectSource 更新源 DC 位置 pbResult 返回结果

HRESULT DrawCurrentLayer([in]OLE_HANDLE hDC,[in]VARIANT_BOOL bDrawChildren,[in]SkinRect rcUpdate,[in]VARIANT_BOOL bMemDC,[out,retval]VARIANT_BOOL *pbResult)
将当前的控件绘制到指定 DC 上
hDC 指定目标 HDC bDrawChildren 是否将控件所有的子控件全部绘制出来

rcUpdate　DC 更新区域 bMemDC 指定 DC 是否为内存 DC

HRESULT MoveTop([in] ISkinObjResBase* pSubObj, [out,retval] VARIANT_BOOL* pbResult)
将某一子控件移动到顶层

HRESULT DrawSprite([in]OLE_HANDLE hDCDest,[in]SkinRect rcDest,[in]SkinRect rcUpdateObject,[in]OLE_HANDLE hFuncDrawBack,[in]OLE_HANDLE pCallClassPtr,[in]VARIANT_BOOL bUpdateCache,[out,retval]VARIANT_BOOL *pbResult)

HRESULT DrawBlindObject([in]OLE_HANDLE hDCDest,[in]SkinRect rcDest,[in]SkinRect rcUpdateObject,[in]SHORT nIndexStart,[in]SHORT nIndexEnd,[out,retval]VARIANT_BOOL *pbResult)

HRESULT DestroyRgnInfo([out,retval]VARIANT_BOOL *pbResult)
销毁控件绘制的区域信息

HRESULT EnableRedraw([in]VARIANT_BOOL bSetRedraw)
关闭控件的重绘，关闭后不会响应 DirectRedraw，例如在 Listview 插入多个 item 前关闭重绘，提高插入速度，插入完毕后重新开启重绘
bSetRedraw 是否开启重绘 VARIANT_TRUE 开启重绘 VARIANT_FALSE 关闭重绘

HRESULT IsEnableRedraw([out,retval]VARIANT_BOOL *bIsSetRedraw)
是否关闭了重绘

HRESULT SetAvailRect([in]IDUIRect* pRect)
获取控件绘制的有效区域

HRESULT GetAvailRect([out,retval]IDUIRect** pRect)
获取控件绘制的有效区域

HRESULT CallEvent([in] LONG nMsgID,[in] LONG wParam,[in] LONG lParam)
调用脚本

4.3 IDUIControlBase ： ISkinObjResBase

HRESULT GetControlTypeName([out,retval]BSTR* pstrResult);
获取控件类型名称
PstrResult 返回类型名称

HRESULT GetAuthorInfo([out]BSTR *pbstrAuthorName,[out]BSTR *pbstrCompany,[out]BSTR *pbstrUrl,[out]BSTR *pbstrVersion);
获取控件的开发者信息
pbstrAuthorName 作者名称 pbstrCompany 作者单位 pbstrURL 作者网站 pbstrVersion 控件版本

HRESULT GetControlIcon([in]int nWidth,[in]int nHeight,[out,retval]OLE_HANDLE* phIcon);
获取控件图标
nMsgID 脚本 ID wparam 参数一 lparam 参数二

HRESULT GetDragCursor([out,retval]OLE_HANDLE *phCursor);
获取控件光标
phCursor 光标句柄

HRESULT SetClassId([in]BSTR clsId);

HRESULT GetClassId([out,retval]BSTR* clsId);

HRESULT GetCategoryName([out,retval]BSTR* pstrResult);
获取控件的分类名称信息 如 KernelAll OfficeAll
pstrResult 返回类型名称

DrawObject([in]IDUIObjectDraw *pObjDraw,[in]SkinRect sknrc,[out,retval]VARIANT_BOOL* pbResult);
绘制控件
pObjDraw sknrc pbResult 返回调用结果

HRESULT DestroyBmpPerPixel();
删除控件缓存

HRESULT Clone([in]IDUIObj *pObjParent,[out,retval]IDUIObj **ppResult);
拷贝控件
pObjParent 拷贝后控件的父控件指针 ppResult 返回拷贝的控件

HRESULT OnMouseEnter([in]SHORT nHitTest,[in]SHORT x,[in]SHORT y,[out,retval]LONG* plResult);
向控件发送 DUISM_MOUSEENTER 消息

HRESULT OnMouseMove([in]SHORT nHitTest,[in]SHORT x,[in]SHORT y,[out,retval]LONG* plResult);
向控件发送 DUISM_MOUSEMOVE 消息

HRESULT OnLButtonDown([in]SHORT nHitTest,[in]SHORT x,[in]SHORT y,[out,retval]LONG* plResult);
向控件发送 DUISM_LBUTTONDOWN 消息

HRESULT OnLButtonDbClick([in]SHORT nHitTest,[in]SHORT x,[in]SHORT y,[out,retval]LONG* plResult);
向控件发送 DUISM_LBUTTONDBCLICK 消息

HRESULT OnLButtonUp([in]SHORT nHitTest,[in]SHORT x,[in]SHORT y,[out,retval]LONG* plResult);
向控件发送 DUISM_LBUTTONUP 消息

HRESULT OnMouseLeave([out,retval]LONG* plResult);
向控件发送 DUISM_MOUSELEAVE 消息

HRESULT OnSetFocus([in]SHORT x,[in]SHORT y,[out,retval]LONG* plResult);
向控件发送 DUISM_SETFOCUS 消息

HRESULT OnKillFocus();
向控件上发送 DUISM_KILLFOUCS 消息

HRESULT OnSize();
向控件发送 DUISM_ONSIZE 消息

HRESULT OnTimer(UINT nIDEvent);
向控件发送 DUISM_TIMER 消息

HRESULT DirectRedraw();
重绘控件

HRESULT CallProp([in]long lPropId,[out,retval]BSTR* pbstrResult);
查询属性

HRESULT AddProp(ICtrlPluginProp* pProp);
添加属性

HRESULT GetPropList(long *pPropList);
获取属性列表

HRESULT Load();

HRESULT AddPlugInObj([in]IDUIControlBase* pPluginObj);

HRESULT RemovePlugInObj([in]IDUIControlBase* pPluginObj);
调用脚本

HRESULT DesignStatusChanged([in]VARIANT_BOOL bDesign);
修改控件是否处于设计模式下

HRESULT CreateObj([in]long pSkinResource,[in]long pObjParent,[in]long nType,[in,defaultvalue(0)]VARIANT_BOOL bInsertFirst,[out,retval]long* pNewObj);
创建对象

HRESULT DrawPreview([in]long hDC,[in]SkinRect *sknrc);
绘制控件预览

HRESULT SetRealCtrlPtr([in]IDUIControlBase* pCtrlBase);
设置

HRESULT GetRealCtrlPtr([out,retval]IDUIControlBase** ppCtrlBase);
获得控件的 controlbase 对象指针

HRESULT SetCursorPtr([in]IDUICursor* pCursor);
设置光标

HRESULT GetCursorPtr([out,retval]IDUICursor** ppCursor);
获取光标

HRESULT SetHotkeyPtr([in]IDUIHotkey *pHotkey1,[in]IDUIHotkey *pHotkey2);
设置快捷键

HRESULT GetHotkeyPtr([out]IDUIHotkey **ppHotkey1,[out]IDUIHotkey **ppHotkey2);
获取快捷键

HRESULT SetBackEnterHotkey([in]VARIANT_BOOL bIsBackSpace,[in]VARIANT_BOOL bIsEnter);

HRESULT GetBackEnterHotkey([out]VARIANT_BOOL *pbIsBackSpace,[out]VARIANT_BOOL *pbIsEnter);

HRESULT CreateGroupProp([in]ICtrlPluginProp *pParentProp,[in]BSTR strPropName,[in]VARIANT_BOOL bExpand,[out,retval]ICtrlPluginProp **ppProp);
创建属性组，通常在创建

HRESULT CreateProp([in]ICtrlPluginProp* pPropParent,[in]enumPropType eType,[in]BSTR strPropName,[in]BSTR strPropHelp,[in]VARIANT_BOOL bIsStyle,[in,defaultvalue(0)]IPropChangedCallback *pCallback,[out,retval]IDUIPropBase **ppProp);
创建属性

HRESULT ExportConfig([in]OLE_HANDLE pXmlEle,[out,retval]VARIANT_BOOL *pbResult);
导出控件资源文件

HRESULT ImportConfig([in]IDUIRes *pDUIRes,[in]OLE_HANDLE hObjParent,[out,retval]IDUIControlBase **ppResult);
通过导入的控件资源文件创建控件

HRESULT GetFilePath([out,retval]BSTR *pstrPath);
获取皮肤路径

HRESULT DrawChildren([in]OLE_HANDLE hDC,[in]SkinRect sknrc);
绘制控件所有的子控件

HRESULT IsDragabled([out,retval]VARIANT_BOOL *pbResult);
点击此控件是否可拖拽

HRESULT EventNotify([in]DUINotify *peVentNotify,[out,retval]VARIANT_BOOL *pbResult);
向控件发送事件响应

HRESULT PropChangedNotify([in]IDUIPropBase *pProp);

HRESULT SetRootObject([in]VARIANT_BOOL bRootObj);
设置根对象

HRESULT IsRootObject([out,retval]VARIANT_BOOL *pbResult);
判断是否为根对象

HRESULT CreateProps();
创建属性

HRESULT SetFocus([out,retval]VARIANT_BOOL *pbResult);
设置焦点状态

HRESULT SetTabStop([in]VARIANT_BOOL bTabStop);
设置是否 Tab 切换，方法保留

HRESULT IsTabStop([out,retval]VARIANT_BOOL *pbResult);
是否可 Tab 切换，方法保留

HRESULT SetDrawFocus([in]VARIANT_BOOL bDrawFocus);
设置是否绘制焦点框，方法保留

HRESULT IsDrawFocus([out,retval]VARIANT_BOOL *pbResult);
是否绘制焦点状态，方法保留

HRESULT AddMsgCallback([in]LONG nMsgID,[in]OLE_HANDLE hFuncCallback,[in]OLE_HANDLE pThis,[out,retval]VARIANT_BOOL *pbResult);
添加脚本

HRESULT RemoveMsgCallback([in]LONG nMsgID,[in]OLE_HANDLE hFuncCallback,[out,retval]VARIANT_BOOL *pbResult);
调用脚本

HRESULT GetMsgCallbackCount([in]LONG nMsgID,[out,retval]SHORT *pnResult);

HRESULT GetCallbackAddress([in]LONG nMsgID,[in]OLE_HANDLE pThis,[out,retval]OLE_HANDLE *phResult);

HRESULT CallMsg([in]LONG nMsgID,[in]LONG wParam,[in]LONG lParam,[out,retval]VARIANT_BOOL *pbResult);
调用事件消息

HRESULT RebuildPropBaseXMLNode();
重建属性列表

HRESULT SetIsComposeCtrl([in] VARIANT_BOOL bIsComposeCtrl);
设置是否为组合控件

HRESULT IsComposeCtrl([out, retval] VARIANT_BOOL* bIsComposeCtrl);
查询是否是组合控件

HRESULT ClearPropBaseXMLNode();
清理时间节点

HRESULT DUISetToolTip([in]BSTR strTool);
设置控件的 tooltip
strTool 设置 tooltip

HRESULT DUIGetToolTip([out, retval]BSTR* strTool);
获取控件的 tooltip
strTool 获取 tooltip

HRESULT SetDragabledProp([in] VARIANT_BOOL bDragabled);
设置控件是否支持拖拽数据

HRESULT IsDragabledProp([out,retval] VARIANT_BOOL *pbResult);

是否开启控件可拖拽数据

5. KernelControls 控件接口介绍

5.1. ICmdButton： IDUIControlBase 按钮控件接口说明

HRESULT SetText([in] BSTR strText)
设置按钮上显示的文本
StrText 文本内容

HRESULT GetText([out,retval] BSTR* pstrResult)
获取按钮上的文本
PstrResult 返回的文本

HRESULT GetButtonState([out,retval] DUI_BUTTONSTATE* pResult)
获取按钮的当前状态
pResult 返回当前状态

HRESULT SetButtonState([in] DUI_BUTTONSTATE eState)
设置按钮状态
eState 按钮状态

HRESULT SetGraphicsFilePath([in] BSTR strFilePath, SHORT nOffsetX, [in] SHORT nOffsetY, [in] SHORT nCount, [in] VARIANT_BOOL bHorz);
通过路径设置按钮图标
StrFilePath 图标路径 nOffsetX 水平偏移 nOffsetY 垂直偏移 nCount 图片数量，设置为 1 bHorz 是否是水平的，这是为 VARIANT_TRUE

HRESULT SetGraphicshIcon([in] OLE_HANDLE hIcon, [in] SHORT nOffsetX, [in] SHORT nOffsetY)
通过 HICON 设置按钮的图标
hIcon 图标的句柄 nOffsetX 水平偏移 nOffsetY 垂直偏移

HRESULT GetGraphicshIcon([out,retval] OLE_HANDLE* phIcon)
如果设置了图标 HICON，获得此 HICON
phIcon 获得 HICON

HRESULT SetUpDownMode([in] VARIANT_BOOL bUpDownMode)
设置图标的对齐方式
bUpDownMode VARIANT_TRUE 垂直居中 VARIANT_FALSE 水平居中

HRESULT IsUpDownMode([out,retval] VARIANT_BOOL* pbResult)
获得图标的对齐方式
pbResult 返回对齐方式

HRESULT SetTooltip([in] BSTR strTooltip)
设置按钮的 ToolTip
StrToolTip：tooltip 文本

HRESULT GetTooltip([out,retval] BSTR* pstrResult)
获取按钮上的 tooltip 文本
pstrResult 返回文本

HRESULT SetGraphicOffset([in] SHORT nOffsetX, [in] SHORT nOffsetY);
设置按钮上图标的偏移量

nOffsetX 水平偏移量 nOffsetY 垂直偏移量

HRESULT GetGraphicOffset([in,out] SHORT*, [in,out] SHORT* pnOffsetY)
获取按钮上图标的偏移量
pnOffsetX 返回水平偏移量 pnOffset 返回垂直偏移量

HRESULT ShowText([in] VARIANT_BOOL bShow)
设置按钮是否显示文本
bShow 是否显示文本

HRESULT IsShowText([out,retval] VARIANT_BOOL* pbResult)
按钮是否显示文本
pbResult 是否显示文本

HRESULT SetBackImageSec([in] DUI_BUTTONSTATE eState,[in]IDUIImageBase* pImageBase)
设置按钮的状态背景
eState 按钮状态 pImageBase 状态对应 ImageBase

HRESULT GetBackImageSec([in] DUI_BUTTONSTATE eState,[out,retval]IDUIImageBase** ppImageBase)
获取按钮对应背景 ImageBase
eState 按钮状态 ppImageBase 返回背景 ImageBase

HRESULT SetBackTextStyle([in] DUI_BUTTONSTATE eState,[in]IDUITextStyle* pTextStyle)
设置按钮文本样式
eState 按钮状态 pTextStyle 文字样式

HRESULT GetBackTextStyle([in] DUI_BUTTONSTATE eState,[out,retval]IDUITextStyle** ppTextStyle)
获取按钮的文字样式
eState 按钮状态 ppTextStyle 文字样式

HRESULT SetGraphicsImage([in]DUI_BUTTONSTATE eState,[in]IDUIImageBase* pImageBase, [in]VARIANT_BOOL bRedraw,[out,retval]VARIANT_BOOL* pbResult)
设置按钮的图标背景
eState 按钮状态 ppImageBase 图标背景 bRedraw 设置完成后是否立即重绘控件 pbResult 返回设置结果

HRESULT GetGraphicsImage([in] DUI_BUTTONSTATE eState, [out,retval] IDUIImageBase** pResult)
获取图标 ImageBase
eState 按钮状态 pResult 返回 ImageBase

HRESULT SetFileIcon([in] BSTR sFilePath)
通过路径设置图标
strFilePath 图标路径

HRESULT GetTextLength([out,retval] SHORT* pnResult)
获取按钮文本的长度
pnResult 返回长度

HRESULT SetActive([in]VARIANT_BOOL bActive)
设置按钮是否处于激活状态，此方法已过时，设置控件启用通过 EnableObject 接口来设置
bActive 是否激活

HRESULT IsActive([out,retval]VARIANT_BOOL* pbResult)

获取按钮是否激活
pbResult

HRESULT SetGraphicshBitmap([in] OLE_HANDLE hBitmap,[in] VARIANT_BOOL bRedraw)
通过 HBitmap 设置按钮的图标
hBitmap 图标的 HBitmap bRedraw 设置完成后是否立即重绘

HRESULT GetGraphicshBitmap([out,retval] OLE_HANDLE* phResult)
互殴设置的图标 HBitmap
phResult 返回图标图像句柄

5.2. IDUIAnimate： IDUIControlBase 动画控件接口说明

HRESULT SetFramesCount([in] SHORT nCount)
设置图像分割帧的数量
nCount 设置图像帧数量

HRESULT GetFramesCount([out,retval] SHORT* pnResult)
获取图像帧的数量
pnResult 返回图像帧数量

HRESULT SetSpeed([in] SHORT nSpeed)
设置动画的播放速度
nSpeed 动画速度

HRESULT GetSpeed([out,retval] SHORT* pnResult)
获取动画的播放速度
pnResult 返回速度

HRESULT StartAnimate()

开始动画

HRESULT StopAnimate()
停止动画

HRESULT SetAutoStop([in] VARIANT_BOOL bAutoStop, [in] SHORT nElapse)
设置动画是否自动停止
bAutoStop 是否自动停止 nElapse 播放次数

5.3. IDUICheckBox： IDUIControlBase 多选控件接口说明

HRESULT SetText([in] BSTR strText)
设置控件显示文本
strText 文本内容

HRESULT GetText([out,retval] BSTR* pstrResult)
获取控件显示文本
pstrResult 返回文本

HRESULT GetButtonState([out,retval] DUI_CHECKBOXSTATE* pResult)
获取控件状态 正常、高亮、按下、禁用
pResult 返回当前状态

HRESULT SetButtonState([in] DUI_CHECKBOXSTATE eState)
设置控件状态
eState 设置控件状态

HRESULT GetValue([out,retval] DUICHECKBOX_VALUE* pResult)
查询控件是否被选取
pResult 返回参数

HRESULT SetValue([in] DUICHECKBOX_VALUE eValue)
设置控件是否被选取
eValue 选中 未选取 半选

HRESULT SetThreeState([in] VARIANT_BOOL bThreeState)
是否开启半选功能
bTreeState 是否开启半选

HRESULT GetThreeState([out,retval] VARIANT_BOOL* pbResult)
查询是否开启半选
pbResult 返回是否半选开启

HRESULT SetTooltip([in] BSTR strTooltip)
设置控件 ToolTip
strToolTip ToolTip 文本内容

HRESULT GetTooltip([out,retval] BSTR* pstrResult)
查询控件 tooltip
pstrResult 返回 tooltip 文本

HRESULT ShowText([in] VARIANT_BOOL bShow)
设置控件是否显示文本
bShow 是否显示文本

HRESULT IsShowText([out,retval] VARIANT_BOOL* pbResult)
查询是否显示文本

pbResult 返回结果

HRESULT SetPushButtonLike([in]VARIANT_BOOL bPushButtonLike)
设置控件是否是 Button 样式
bPushButtonLike Button 样式

HRESULT IsPushButtonLike([out,retval]VARIANT_BOOL *pbResult)
查询是否为 Button 样式
pbResult 返回结果

HRESULT SetActive([in]VARIANT_BOOL bActive)
设置控件是否激活，已过时，请使用 EnableObject 方法

HRESULT IsActive([out,retval]VARIANT_BOOL* pbResult)
查询控件是否已激活，已过时，请使用 IsEnableObject 方法

HRESULT SetData([in]OLE_HANDLE hData)
为此控件设置一个附加数据
hData 附加数据内容更

HRESULT GetData([out,retval]OLE_HANDLE* phData)
获取控件绑定的附加数据
phData 返回数据内容

5.4. IUIForm： IDUIControlBase 背景控件接口说明

HRESULT SetDrawColor([in]VARIANT_BOOL bDrawColor)

设置使用使用颜色来绘制背景
bDrawColor 是否使用颜色绘制

HRESULT GetDrawColor([out,retval]VARIANT_BOOL *pbDrawColor)
查询是否通过颜色来绘制背景
pbDrawColor 返回查询结果

HRESULT SetBackImageSec([in]IDUIImageBase* pImageBase)
设置背景图片
pImageBase 背景图片

HRESULT GetBackImageSec([out,retval]IDUIImageBase** ppImageBase)
获取背景图片
ppImageBase 返回背景图片

HRESULT SetBackColor([in]IFillColor* pFillColor)
设置背景填充颜色
pFillColor 填充的颜色

HRESULT GetBackColor([out,retval]IFillColor** ppFillColor)
获取填充颜色
ppFillColor 返回颜色

HRESULT SetBackBitmapPath([in]BSTR szImagePath, [in]VARIANT_BOOL bRedraw,[out,retval]VARIANT_BOOL* pbResult)
通过路径设置背景
szImagePath 图片路径 bRedraw 设置后是否重绘 pbResult 返回设置结果

HRESULT GetBackBitmapPath([out,retval]BSTR* pszImagePath)
获取设置的背景图片路径

pszImagePath 返回图片路径

HRESULT SetShowBitmap([in]VARIANT_BOOL bShowBitmap)
设置是否显示背景图片
phData 是否显示

HRESULT IsShowBitmap([out,retval]VARIANT_BOOL *pbShowBitmap)
查询是否显示 Bitmap 图片背景
pbShowBitmap 返回结果

5.5. IRadioBox： IDUIControlBase 单选控件接口说明

HRESULT SetText([in] BSTR strText,[in]VARIANT_BOOL bRedraw)
设置控件文本
strText 文本内容 bRedraw 设置后是否立即重绘

HRESULT GetText([out,retval] BSTR* pstrResult)
获取控件文本
pbShowBitmap 返回结果

HRESULT GetButtonState([out,retval] DUI_RADIOBOXSTATE* pResult)
查询控件状态
pbShowBitmap 返回结果

HRESULT SetButtonState([in] DUI_RADIOBOXSTATE eState,[in]VARIANT_BOOL bRedraw)
设置控件状态
eState 控件状态 bRedraw 设置后是否立即重绘

HRESULT SetGraphicsFilePath([in] BSTR strFilePath, SHORT nOffsetX, [in] SHORT nOffsetY, [in] SHORT

nCount, [in] VARIANT_BOOL bHorz)
通过图片路径来设置图标
strFilePath 图标路径 nOffsetX 水平偏移 nOffsetY 垂直偏移 nCount 图片数量，设置为 1 bHorz 是否为水平，设置为 VARIANT_FALSE

HRESULT GetGraphicsFilePath([out,retval] BSTR* pstrResult)
查询设置的图标路径
pstrResult 返回路径

HRESULT SetValue([in] DUIRADIOBOX_VALUE eValue,[in]VARIANT_BOOL bRedraw)
设置控件是否被选中
eValue 是否选中 bRedraw 设置后是否立即重绘

HRESULT GetValue([out,retval] DUIRADIOBOX_VALUE* pResult)
查询控件是否被选中
pbShowBitmap 返回结果

HRESULT SetUpDownMode([in] VARIANT_BOOL bUpDownMode,[in]VARIANT_BOOL bRedraw)
设置图标的对齐方式
bUpDownMode VARIANT_TRUE 则垂直居中对齐 VARIANT_FALSE 垂直居中 bRedraw 设置完毕后是否立即重绘

HRESULT IsUpDownMode([out,retval] VARIANT_BOOL* pbResult)
查询图标的对齐方式
pbResult 返回图标的对齐方式

HRESULT SetTooltip([in] BSTR strTooltip)
设置控件的 tooltip
strTooltip 控件的 tooltip 文本

HRESULT GetTooltip([out,retval] BSTR* pstrResult)
查询控件的 tooltip
pstrResult 返回文本

HRESULT SetGraphicOffset([in] SHORT nOffsetX, [in] SHORT nOffsetY,[in]VARIANT_BOOL bRedraw)
设置图标的偏移量
nOffsetX 水平偏移量 nOffsetY 垂直偏移量 bRedraw 设置后是否立即重绘

HRESULT GetGraphicOffset([in,out] SHORT* pnOffsetX, [in,out] SHORT* pnOffsetY)
获取图标的偏移量
pnOffsetX 水平偏移量 pnOffsetY 垂直偏移量

HRESULT ShowText([in] VARIANT_BOOL bShow,[in]VARIANT_BOOL bRedraw)
设置是否显示文本
bShow 是否显示文本 bRedraw 设置后是否立即重绘

HRESULT IsShowText([out,retval] VARIANT_BOOL* pbResult)
查询是否显示文本
pbResult 返回结果

HRESULT StartBlink([in] SHORT nIDEvent, [in] SHORT nElapse)
设置控件开始闪烁
nIdEvent 时间 ID nElapse 时间间隔

HRESULT StopBlink(void)
停止闪烁

HRESULT SetChildWndImage([in] BSTR strPicPath, [in] SHORT nRightSpaceX, [in] SHORT nOffsetY)

HRESULT SetBlinkImage([in] BSTR strPicPath)
设置闪烁图片的路径
strPicPath 闪烁图片的路径

HRESULT SetBackImage([in]DUI_RADIOBOXSTATE eState,[in]VARIANT_BOOL bChecked,[in]IDUIImageBase *pImageBase,[out,retval]VARIANT_BOOL *pbResult)
设置控件的背景图片
eState 控件的需要设置的状态 bChecked 是否是选取状态图片 pImageBase 设置的图片 pbResult 返回设置结果

HRESULT GetBackImage([in]DUI_RADIOBOXSTATE eState,[in]VARIANT_BOOL bChecked,[out,retval]IDUIImageBase **ppImageBase)
查询背景图片
eState 需要查询的控件状态 bChecked 是否是选取状态图片 ppImageBase 返回图片

HRESULT SetBackColor([in]DUI_RADIOBOXSTATE eState,[in]VARIANT_BOOL bChecked,[in]OLE_COLOR clrBack,[out,retval]VARIANT_BOOL *pbResult)
设置控件背景颜色
eState 控件的需要设置的状态 bChecked 是否是选取状态 clrBack 设置的颜色 pbResult 返回设置结果

HRESULT GetBackColor([in]DUI_RADIOBOXSTATE eState,[in]VARIANT_BOOL bChecked,[out,retval]OLE_COLOR *pClrBack)

查询设置的颜色
eState 需要查询的控件状态 bChecked 是否是选取状态图片 pClrBack 返回的控件颜色

HRESULT SetBoxImage([in]DUI_RADIOBOXSTATE eState,[in]VARIANT_BOOL bChecked,[in]IDUIImageBase *pImageBase,[out,retval]VARIANT_BOOL *pbResult)
设置控件图标图片
eState 控件的需要设置的状态 bChecked 是否是选取状态 clrBack 设置的颜色 pbResult 返回设置结果

HRESULT GetBoxImage([in]DUI_RADIOBOXSTATE eState,[in]VARIANT_BOOL bChecked,[out,retval]IDUIImageBase **ppImageBase)
获取控件图标图片
eState 控件的需要设置的状态 bChecked 是否是选取状态 ppImgeBase 图标图片

HRESULT SetGraphicsImage([in]DUI_RADIOBOXSTATE eState,[in]VARIANT_BOOL bChecked,[in]IDUIImageBase *pImageBase,[out,retval]VARIANT_BOOL *pbResult)
查询是否显示 Bitmap 图片背景
eState 控件的需要设置的状态 bChecked 是否是选取状态 ppImgeBase 图标图片 pbResult 返回设置结果

HRESULT GetGraphicsImage([in]DUI_RADIOBOXSTATE eState,[in]VARIANT_BOOL bChecked,[out,retval]IDUIImageBase **ppImageBase)
查询是否显示 Bitmap 图片背景
pbShowBitmap 返回结果

HRESULT SetGraphicsBmp([in]DUI_RADIOBOXSTATE eState,[in]VARIANT_BOOL bChecked,[in]OLE_HANDLE hBitmap,[out,retval]VARIANT_BOOL *pbResult)

通过 HBITMAP 来设置图标
eState 需要设置的状态 bChecked 是否是选取状态 hBitmap 传入 hbtimap pbResult 返回设置结果

HRESULT GetGraphicsBmp([in]DUI_RADIOBOXSTATE eState,[in]VARIANT_BOOL bChecked,[out,retval]OLE_HANDLE *phBitmap)
获取设置的图标 hbtimap
eState 需要获取的状态 bChecked 是否是选取状态 phBitmap 返回 hbtimap

HRESULT SetTextStyle([in]DUI_RADIOBOXSTATE eState,[in]VARIANT_BOOL bChecked,[in]IDUITextStyle *pTextStyle,[out,retval]VARIANT_BOOL *pbResult)
设置文本样式
eState 需要设置的状态 bChecked 是否是选取状态 pTextStyle 文本样式 pbResult 返回设置结果

HRESULT GetTextStyle([in]DUI_RADIOBOXSTATE eState,[in]VARIANT_BOOL bChecked,[out,retval]IDUITextStyle **ppTextStyle)
查询文本样式
eState 需要设置的状态 bChecked 是否是选取状态 pTextStyle 文本样式

HRESULT SetBackDrawColor([in]VARIANT_BOOL bDrawColor)
设置使用图片或者颜色来绘制背景
bDrawColor 是否使用颜色绘制

HRESULT GetBackDrawColor([out,retval]VARIANT_BOOL *pbDrawColor)
查询是否使用颜色绘制背景
pb

HRESULT SetData([in]OLE_HANDLE hData)
查询是否显示 Bitmap 图片背景
pbShowBitmap 返回结果

HRESULT GetData([out,retval]OLE_HANDLE *phData)
获取控件的绑定数据
phData 查询控件的绑定数据

HRESULT SetGraphicSize([in]SHORT nWidth,[in]SHORT nHeight,[in]VARIANT_BOOL bRedraw,[out,retval]VARIANT_BOOL *pbResult)
设置图标的大小
nWidht 宽度 nHeight 高度 bRedraw 是否立即重绘 pbResult 返回设置结果

HRESULT GetGraphicSize([out]SHORT* pnWidth,[out]SHORT* pnHeight)
获取图标的大小
pnWidth 宽度 pnHeight 高度

HRESULT SetGroupID([in] LONG nID,[out,retval] VARIANT_BOOL* pbResult)
设置 radio 组的 id 好
nID 组的 id pbResult 返回设置结果

HRESULT GetGroupID([in] LONG* plResult)
获取组的 id
plResult 返回组 id

HRESULT GetGroup([out,retval] IDUIRadioGroup** ppResult)
获取 Radio 组，已过时

5.6. IDUIHwndObj： IDUIControlBase 子窗口容器接口说明

HRESULT GetSafeHwnd([in] SHORT nIndex,[out,retval] OLE_HANDLE* phResult)
获得指定的与 HwndObj 绑定的窗口句柄
nIndex　要得到的窗口句柄的序号.

HRESULT IsExist([in] OLE_HANDLE hWnd,[out,retval] VARIANT_BOOL* pbResult)
判断窗口是否已经与 HwndObj 绑定。
hWnd 要判断的窗口的窗口句柄. pbResult 返回设置结果

HRESULT Attach([in] OLE_HANDLE hWnd,[out,retval] VARIANT_BOOL* pbResult)
设置窗口与 HwndObj 绑定。
hWnd 要与 HwndObj 绑定的窗口句柄。 pbResult 返回设置结果

HRESULT Detach([in] OLE_HANDLE hWnd,[out,retval] VARIANT_BOOL* pbResult)
解除窗口与 HwndObj 的绑定.
hWnd 要解除绑定的窗口句柄。 pbResult 返回设置结果

HRESULT MoveHwndObjCtrls([in] SkinRect rect,[in] VARIANT_BOOL bPopup)
移动与 HwndObj 绑定的所有窗口的位置.
rect　绑定窗口要移到的位置。 bPopup 该参数是预留参数.

HRESULT ShowWindow([in] OLE_HANDLE hWnd,[out,retval] VARIANT_BOOL* pbResult)
根据窗口句柄指定 Hwndobj 绑定窗口中的窗口显示.
hWnd 指定要显示窗口的窗口句柄.

pbResult 返回设置

HRESULT ShowWindowByIndex([in] SHORT nIndex,[out,retval] VARIANT_BOOL* pbResult)
根据 Hwndobj 绑定窗口中的序号指定窗口显示
nIndex 要显示窗口在 Hwndobj 绑定窗口中的序号 pbResult 返回设置结果

HRESULT GetCurHwnd([out,retval] OLE_HANDLE* phResult)
获得 Hwndobj 绑定窗口中当前显示的窗口句柄
phResult 成功返回当前显示窗口的窗口句柄，否则返回空

HRESULT HideCurWindow([out,retval] VARIANT_BOOL* pbResult)
隐藏 Hwndobj 绑定窗口中当前显示的窗口
PbResult 返回设置结果

HRESULT ShowCurWindow([out,retval] VARIANT_BOOL* pbResult)
显示 HideCurWindow 隐藏的窗口
PbResult 返回设置结果

HRESULT SetClipIntersect([in] VARIANT_BOOL bClip)
设置剪切窗口与 Hwndobj 区域相交部分.
bClip 是否剪切相交部分。

HRESULT IsClipIntersect([out,retval] VARIANT_BOOL* pbResult)
获得是否剪切窗口与 Hwndobj 区域相交部分。
pbResult 返回设置结果

5.7. IDUIListView： IDUIControlBase 列表控件接口说明

HRESULT SetListType([in] DUILV_STYLE eListType,[in] VARIANT_BOOL bRedraw,VARIANT_BOOL *pbResult)

设置 Listview 类型
eListType 类型枚举值 pbResult 返回设置结果

HRESULT GetListType([out,retval] DUILV_STYLE* peListType)
获取 ListView 类型。
peListType 返回类型枚举

HRESULT SetDrawBackColor([in] VARIANT_BOOL bDrawColor,[in] VARIANT_BOOL bRedraw,[out,retval] VARIANT_BOOL *pbResult)
设置绘制背景的方式
bDrawColor 是否通过颜色来绘制背景 bRedraw 设置后是否重绘 pbResult 返回设置结果

HRESULT IsDrawBackColor([out,retval] VARIANT_BOOL *pbResult)
查询背景是否通过颜色绘制
pbResult 返回结果

HRESULT SetBackImage([in] IDUIImageBase* pImageBase,[in] VARIANT_BOOL bRedraw,[out,retval] VARIANT_BOOL *pbResult)
设置控件背景图片
pImageBase 背景图片 bRedraw 设置后是否重绘 pbResult 返回设置结果

HRESULT GetBackImage([out,retval] IDUIImageBase** ppImageBase)
获取控件背景
ppImageBack 返回控件背景

HRESULT SetBackColor([in] IFillColor* pFillColor,[in] VARIANT_BOOL bRedraw,[out,retval] VARIANT_BOOL *pbResult)
设置背景绘制颜色

pFillColor 背景颜色 bRedraw 设置后是否立即重绘 pbResult 返回设置结果

HRESULT GetBackColor([out,retval] IFillColor** ppFillColor)
获取背景颜色
ppFillColor 返回背景颜色

HRESULT SetDrawItemColor([in] VARIANT_BOOL bDrawColor,[in] VARIANT_BOOL bRedraw,[out,retval] VARIANT_BOOL *pbResult)
设置是否用颜色绘制 item 的背景
bDrawColor 是否使用颜色 bRedraw 是否立即重绘 pbResult 返回设置结果

HRESULT IsDrawItemColor([out,retval] VARIANT_BOOL *pbResult)
查询是否使用颜色来绘制 item 背景
pbResult 返回查询结果

HRESULT SetItemHeight([in] SHORT nValue,[in] VARIANT_BOOL bRedraw,[out,retval] VARIANT_BOOL *pbResult)
设置 item 的高度
nValue item 的高度 bRedraw 是否立即重绘 pbResult 返回设置结果

HRESULT GetItemHeight([out,retval] SHORT *pnResult)
查询 Item 的高度
pnResult 返回查询结果

HRESULT SetItemWidth([in] SHORT nValue,[in] VARIANT_BOOL bRedraw,[out,retval] VARIANT_BOOL *pbResult)
设置 item 的宽度
nValue 宽度

bRedraw 是否立即重绘 pbResult 返回设置结果

HRESULT GetItemWidth([out,retval] SHORT *pnResult)
查询 item 的宽度
pnResult 返回 item 宽度

HRESULT GetItemDefImage([in] DUILVI_STATE eState,[out,retval] IDUIImageBase** ppImageBae)
查询 item 某一状态下的背景
eState item 状态 ppImagteBase 返回背景图片

HRESULT SetItemDefImage([in] DUILVI_STATE eState,[out,retval] IDUIImageBase* pImageBae)
设置 item 某一个状态下的背景
eState item 状态 pImageBase 背景图片

HRESULT SetItemDefColor([in] DUILVI_STATE eState,[in] IFillColor* pFillColor,[in] VARIANT_BOOL bRedraw,[out,retval] VARIANT_BOOL* pbResult)
设置 item 的背景颜色
eState item 状态 pFillColor item 颜色 bRedraw 是否立即重绘 pbResult 返回设置结果

HRESULT GetItemDefColor([in] DUILVI_STATE eState,[out,retval] IFillColor** ppFillcolor)
获取 Item 的背景颜色
eState item 状态 ppFillColor item 填充颜色

HRESULT SetTextStyle([in] DUILVI_STATE eState,[in] IDUITextStyle* pTextStyle,[in] VARIANT_BOOL bRedraw,[out,retval] VARIANT_BOOL* pbResult)
设置 item 字体
eState item 状态

pTextStyle 文字样式 bRedraw 设置后是否立即重绘 pbResult 返回设置结果

HRESULT GetTextStyle([in] DUILVI_STATE eState,[out,retval] IDUITextStyle** ppTextStyle)
查询 item 文字样式
eState item 状态 ppText

HRESULT SetItemSpace([in] SHORT nValue,[in] VARIANT_BOOL bRedraw,[out,retval] VARIANT_BOOL *pbResult)
设置 Item 之间的间距
nValue 间距 bRedraw 设置后是否立即重绘 pbResult 返回设置结果

HRESULT GetItemSpace([out,retval] SHORT *pnResult)
查询 Item 的间距
pnResult 返回间距

HRESULT SetScrollSize([in] LONG nSize)
设置滚动条的宽度
nSize 滚动条宽度

HRESULT GetScrollSize([out,retval] LONG *pResult)
获得是否剪切窗口与 Hwndobj 区域相交部分。
pbResult 返回设置结果

HRESULT SetLeftScroll([in] VARIANT_BOOL bLeftScroll)
设置滚动条是否靠左显示
bLeftScroll 是否靠左

HRESULT IsLeftScroll([out,retval] VARIANT_BOOL *pbResult)

查询是否靠左显示
pbResult 返回结果

HRESULT InsertItem([in] SHORT nItem,[in] BSTR szItem,[in] VARIANT_BOOL bRefresh,[out,retval] SHORT* pnResult)
插入一个 Item
nItem Item 的插入位置 从 0 开始 szItem item 的文本 bRefresh 插入后是否立即刷新 pnResult 返回插入位置

HRESULT GetObject([in] BSTR strParent,[in] BSTR strName,[out,retval] IDUIControlBase** ppResult)
自定义 listview 获取其中的
strParent item 的名曾

HRESULT SetListModel([in] IDUIControlBase* pUIForm)
自定义模式下，设置 item 的模板
pUIForm UIForm 控件指针

HRESULT SetItemImage([in] SHORT nItem,[in] DUILVI_STATE eState,[in] IDUIImageBase* pImageBase,[in] VARIANT_BOOL bRedraw,[out,retval] VARIANT_BOOL* pbResult)
设置 item 的背景
nItem item 的序号，从 0 开始 eState 需要设置的 Item 的状态 pImageBase 设置的图片 bRedraw 设置后是否立即重绘 pbResult 返回设置结果

HRESULT GetItemImage([in] SHORT nItem,[in] DUILVI_STATE eState,[out,retval] IDUIImageBase** ppImageBae)
查询 item 的背景图片
nItem item 的序号 eState 需要查询的状态 ppImageBase 返回图片对象指针

HRESULT SetItemColor([in] SHORT nItem,[in] DUILVI_STATE eState,[in] IFillColor* pFillColor,[in] VARIANT_BOOL bRedraw,[out,retval] VARIANT_BOOL* pbResult)
设置 item 的颜色
nItem item 的序号 eState item 的状态 pFillColor 填充颜色 bRedraw 设置后是否立即重绘 pbResult 返回设置结果

HRESULT GetItemColor([in] SHORT nItem,[in] DUILVI_STATE eState,[out,retval] IFillColor** ppFillcolor)
查询 Item 的颜色
nItem item 的序号，从 0 开始 eState Item 的状态 ppFillcolor 返回查询的颜色

HRESULT GetItemCount([out,retval] SHORT* pnResult)
查询插入的项数
pnResult 返回查询结果

HRESULT DeleteItem([in] SHORT nItem)
根据序号删除 item
nItem 需要删除 item 的序号，从 0 开始

HRESULT DeleteAllItems()
删除所有的 item

HRESULT SetItemText([in] SHORT nItem,[in] SHORT iSubItem,[in] BSTR szItem,[in] VARIANT_BOOL bRedraw,[out,retval] VARIANT_BOOL* pbResult)
设置 item 的文本内容
nItem item 的序号 从 0 开始 iSubItem 保留参数，设置为 0 szItem item 的文本 bRedraw 设置后是否立即重绘 pbResult 返回设置结果

HRESULT GetItemText([in] SHORT nItem,[in] SHORT iSubItem, [out,retval] BSTR* pstrResult)
查询 item 的文本
nItem item 的序号，从 0 开始 iSubItem 保留参数，设置为 0 pstrResult 返回结果

SetItemData([in] SHORT nItem,[in] OLE_HANDLE hData,[out,retval] VARIANT_BOOL* pbResult)
为 Item 设置一块额外的数据
nItem item 的序号，从 0 开始 hData 额外的数据 pbResult 返回设置结果

HRESULT GetItemData([in] SHORT nItem,[out,retval] OLE_HANDLE* phData)
查询 item 的额外数据
nItem item 的序号 phData 返回额外的数据

HRESULT GetItemRect([in] SHORT nItem,[out,retval] SkinRect* rect)
查询 item 的范围
nItem item 的序号，从 0 开始 rect 返回查询的范围

HRESULT SetItemGraphic([in]SHORT nItem,[in] DUILVI_STATE eState,[in] OLE_HANDLE hBitmap,[out,retval] VARIANT_BOOL* pbResult)
设置 item 的图标
nItem item 的序号 eState item 的状态 hBitmap 图标的 hBitmap 句柄 pbResult 返回设置结果

HRESULT GetItemGraphic([in] SHORT nItem,[in] DUILVI_STATE eState,[out,retval] OLE_HANDLE* phResult)
获取设置的 item 图标句柄
nItem item 的序号

eState item 的状态 phResult 返回查询结果

HRESULT SetControlID([in] SHORT nIndex)
设置控件的 id 号，已过时
nIndex 控件 id

HRESULT GetControlID([out,retval] SHORT * pnResult)
查询控件的 id 号
pnResult 返回查询结果

HRESULT SetUseScrollBar([in] VARIANT_BOOL bUseScrollBar)
设置是否使用滚动条
bUseScrollBar 是否使用滚动条

HRESULT IsUserScrollBar([out,retval] VARIANT_BOOL *pbResult)
查询是否使用滚动条
pbResult 返回查询结果

HRESULT RefreshView()
刷新 listview，调整大小和刷新显示

HRESULT GetResObject([in] SHORT nItem, [in] BSTR strName, [out,retval] IDUIControlBase** pItemPoint)
自定义模式下获取 item 里面的 uiform 中的子控件
nItem item 的序号 strName 模板中的控件名称 pItemPoint 返回控件指针

HRESULT GetHeaderCtrl([out,retval] IDUIControlBase** ppResult)
Report 样式下，获取 HeaderCtrl 控件指针
ppResult 返回控件指针

HRESULT InsertColumn([in] SHORT nCol,[in] BSTR strText,[in] SHORT nWidth,[out,retval] IDUILVColumn** ppResult)
Report 样式下，插入新的一列
nCol 列插入的位置，从 0 开始 strText 列的标题 nWidht 列的宽度 ppResult 返回设置的结果

HRESULT DeleteColumn([in] SHORT nCol,[out,retval] VARIANT_BOOL* pbResult)
Report 样式下，删除一列
nCol 删除的列的序号，从 0 开始 pbResult 返回删除结果

HRESULT GetColumn([in] SHORT nCol,[out,retval] IDUILVColumn** ppResult)
Report 样式下，获取列对象
nCol 列的序号，从 0 开始 ppResult 返回列对象

HRESULT IsDrawGrid([out,retval] VARIANT_BOOL* bDrawGrid)
查询 report 样式下是否绘制单元表格线
bDrawGrid 返回查询结果

HRESULT SetDrawGrid([in]VARIANT_BOOL bDrawGrid)
Report 样式下设置是否绘制单元表格
bDrawGrid 是否绘制单元表格线

HRESULT AppendItemText([in]SHORT nItem, [in]SHORT iSubItem, [in]BSTR szItem, [in]BSTR szURL, [in]VARIANT_BOOL bRedraw,[out,retval]VARIANT_BOOL* pbResult)
Report 样式下，插入单元格数据
nItem item 的序号，从 0 开始 iSubItem 列序号，从 0 开始 szItem item 的文本内容 szURL item rul 的文本内容 bRedraw 插入后是否重绘

pbResult 返回结果

HRESULT AppendImage([in]SHORT nItem, [in]SHORT iSubItem, [in]BSTR szImagePath,[in] VARIANT_BOOL bRedraw,[out,retval] VARIANT_BOOL* pbResult)
Report 样式下，向单元格插入图片
nItem item 的序号，从 0 开始 iSubItem 列序号，从 0 开始 szImagePath 图片的路径 bRedraw 插入后是否重绘 pbResult 返回结果

HRESULT GetUnitItemStaticTextStyle([in]DUILVI_STATE eState,[out,retval]IDUITextStyle** ppTextStyle)
Report 样式下，查询单元格的文字样式
eState 状态 ppTextStyle 返回文字样式

HRESULT GetUnitItemUrlTextStyle([in]DUILVI_STATE eState,[out,retval]IDUITextStyle** ppTextStyle)
获取单元格 URL 文字样式
eState 状态 ppTextStyle 返回文字样式

HRESULT SetUnitItemStaticTextStyle([in]DUILVI_STATE eState,[in]IDUITextStyle* ppTextStyle,[in]VARIANT_BOOL bRedraw,[out,retval] VARIANT_BOOL* pbResult)
设置单元格文字样式
eState 状态 ppTextStyle 文字样式 bRedraw 是否重绘 pbResult 返回设置结果

HRESULT GetUnitItem([in]SHORT nRow,[in]SHORT nCol,[out, retval]IDUIUnitItem** ppResult)
Report 样式下，获取单元格对象
nRow 行序号，从 0 开始 nCol 列序号，从 0 开始 ppResult 返回的单元格对象

HRESULT SetUnitItemUrlTextStyle([in]DUILVI_STATE eState,[in]IDUITextStyle* ppTextStyle,[in]VARIANT_BOOL bRedraw,[out,retval] VARIANT_BOOL* pbResult)
设置单元格 URL 文本的文本样式
eState item 状态 ppTextStyle 文字样式 bRedraw 设置后是否重绘 pbResult 返回设置结果

HRESULT GetSelItem([out,retval] IDUITVItemBase** ppResult)
获取选择的 item 对象指针
pbResult 返回控件对象

HRESULT AddGroup([in]LONG nID,[in]BSTR strName,[out,retval]IDUITVItemGroup** ppResult)
添加组，需要皮肤里面开发组的支持
nId 组的 id strName 组的名称 ppResult 返回组对象指针

HRESULT RemoveGroupByID([in]LONG nID,[out,retval]VARIANT_BOOL* pbResult)
通过 Id 号来删除组
nId 组的 id pbResult 返回删除结果

HRESULT RemoveGroupByName([in]BSTR strName,[out,retval]VARIANT_BOOL* pbResult)
通过组的名称来删除组
strName 组的名称 pbResult 返回删除结果

HRESULT RemoveAllGroup()
删除所有的组

HRESULT GetGroupByName([in]BSTR strName,[out,retval]IDUITVItemGroup** ppResult)

通过名称来获取组对象指针
strName 组名称 ppResult 返回组的对象指针

HRESULT GetGroupByID([in]LONG nID,[out,retval]IDUITVItemGroup** ppResult)
通过组的 id 来获取组对象指针
ppResult 返回组对象指针

HRESULT GetGroupCount([out,retval]LONG* pnResult)
查询组的数量
pnResult 返回数量

HRESULT SetSupportMultiSel([in] VARIANT_BOOL bMutilSel)
设置是否支持多选
bMutilSel 是否开启多选功能

HRESULT IsSupportMultiSel([out,retval] VARIANT_BOOL* pbResult)
查询是否开启了多选
pbResult 是否开启了多选

HRESULT GetSelItemCount([in] SHORT* pnResult)
获取已选择 item 的数量
pnResult 返回数量

HRESULT GetFirstSelItem([out,retval] IDUITVItemBase** ppResult)
获得第一个选择到的 item
ppResult 返回 item 指针

HRESULT GetNextSelItem([in]IDUITVItemBase* pItem,[out,retval]IDUITVItemBase** ppResult)
返回下一个选择到的 item 指针，用来遍历多选时使用
pItem 当前 item ppResult 当前 item 的下一个 item

HRESULT InsertGroup([in]SHORT nIndex,[in]LONG nID,[in]BSTR strName,[out,retval]IDUITVItemGroup** ppResult)
插入组
nIndex 插入的位置，从 0 开始 nid 组的 id 号 strName 组的名称 ppResult 返回组的对象指针

HRESULT SortItems([in]OLE_HANDLE lpCmdFun,[in]IDUITVItemGroup* pParentItem,[in]VARIANT_BOOL bRefresh)
对组进行排序
lpCmdFun 排序函数指针 pParentItem 排序某一特定组时传入的组对象指针 bRefresh 排序后是否刷新

HRESULT SetSelItem([in]IDUITVItemBase* pItemBase, [in]VARIANT_BOOL bChangePos)
设置选择的 item
pItemBase 选择的 item bChangePos 选择后是否自动滚动滚动条到选择 item 的位置

HRESULT GetItem([in]BSTR strName,[out,retval]IDUITVItemBase** ppResult)
通过 item 的名称来获取 item 对象指针
strName item 的名称 ppResult 返回 item 对象指针

HRESULT GetAt([in]SHORT nIndex,[out,retval]IDUITVItemBase** ppResult)
通过序号来获取 item
nIndex item 序号，从 0 开始 ppResult 返回 item 对象指针

HRESULT GetListModel([out,retval]IDUIControlBase** ppResult)
自定义模式下获取模板指针
ppResult 返回模板指针

HRESULT GetCheckItemCount([out,retval]SHORT* pnResult)
Report 样式下开启 item checkbox 模式时获取 check 的 item 数量
pnResult 返回选取的数量

HRESULT GetFirstCheckItem([out,retval]IDUITVItem** ppResult)
获取第一个 check 的 item
ppResult 返回第一个 check 的 item 对象指针

HRESULT GetNextCheckItem([in]IDUITVItem* pItem,[out, retval]IDUITVItem** ppResult)
获取下一个 check 的 item
ppItem 当前 check 的 item ppResult 返回当前 check 的 item 的下一个 item

HRESULT SetCheckItem([in]IDUITVItem* pItem,[in]VARIANT_BOOL bCheck)
Check 或者 Uncheck 某一个 item
pItem 需要设置的 item bCheck 是否 check

HRESULT SetColumnUserModule([in]int nCol, [in]IDUIControlBase* pIControlBaseModule, [out,retval]VARIANT_BOOL* pResult)
设置单元格的列模板
nCol 列的 id pIControlBaseModule 模板的对象指针

HRESULT GetColumnUserModule([in]int nCol, [out, retval]IDUIControlBase** pIControlBaseModule)
获取单元格的列模板
nCol 列的序号 pIControlBaseModule 返回控件模板指针

HRESULT GetItemCtrlBase([in]SHORT nRow, [in]SHORT nCol, [out, retval]IDUIControlBase** pIControlBase)
获取单元格内的控件对象指针

nRow 行序号，从 0 开始 nCol 列序号，从 0 开始 plControlBase 返回控件对象指针

HRESULT GetLastVisibleItem([out,retval] IDUITVItemBase** ppResult)
查询最后可见的 item
ppResult 返回 item 指针

HRESULT SetMultiSelItem([in]IDUITVItemBase* pItemBase,[in] VARIANT_BOOL bAutoCancelSel)
获得是否剪切窗口与 Hwndobj 区域相交部分。
pbResult 返回设置结果

HRESULT SetUserModulePosition([in]int nCol, [in]int nOffsetX, [in]int nOffsetY, [in]int nRightMargin, [in]int nHeight)

HRESULT IsClipIntersect([out,retval] VARIANT_BOOL* pbResult)
获得是否剪切窗口与 Hwndobj 区域相交部分。
pbResult 返回设置结果

HRESULT SetAllItemCheck([in]VARIANT_BOOL bCheck)
设置是全 check 或者 uncheck item
bCheck 是否 check

HRESULT SwapCol([in]LONG nIndex1, [in]LONG nIndex2, [out,retval]VARIANT_BOOL* pbResult)
Report 样式下，交换列 1 和列 2
nIndex1 需要交换的列序号 1 nIndex2 需要交换的列序号 2 pbResult 返回设置结果

5.8. IDUILVColumn : IDispath

HRESULT SetColumnWidth([in] SHORT nWidth,[out,retval] VARIANT_BOOL* pbResult)
设置列的宽度
nWidht 列的宽度 pbResult 返回设置结果

HRESULT GetColumnWidth([out,retval] SHORT* pnResult)
获取列的宽度
pnResult 返回宽度值

HRESULT SetAlignMode([in] DUILV_UNITITEM_HORZ eHovAlign,[in] DUILV_UNITITEM_VERT eVerAlign)
设置列内数据的对齐方式
eHovAlign 水平的对齐方式 eVerAlign 垂直的对齐方式

HRESULT GetVerAlignMode([out,retval] DUILV_UNITITEM_VERT* peResult)
获取列内数据的垂直对齐方式
peResult 返回垂直对齐方式

HRESULT GetHorzAlignMode([out,retval] DUILV_UNITITEM_HORZ* peRsult)
获取水平的对齐方式
peResult 返回对齐方式

5.9. IDUITVItemBase : IDispath

HRESULT SetText([in]BSTR strText,[in]VARIANT_BOOL bRedraw)
设置 item 的文本
strText 文本内容 bRedraw 是否重绘

HRESULT GetText([out,retval]BSTR* pstrResult)
获取 item 文本
pstrResult 返回文本

HRESULT SetID([in]LONG nID)

设置 item 的 id
nId 返回 Id

HRESULT GetID([out,retval]LONG* pnResult)
获取 Item 的 Id
pnResult 返回 id

HRESULT SetData([in]OLE_HANDLE hData,[out,retval]VARIANT_BOOL* pbResult)
设置 item 的额外数据
hData 额外的数据 pbResult 返回设置结果

HRESULT GetData([out,retval]OLE_HANDLE* phData)
获取 item 的额外数据
phData 返回额外数据

HRESULT GetType([out,retval]DUILV_TYPE *peResult)
获取 item 的类型，普通的 item 或 group
peResult 返回类型

HRESULT GetRect([out,retval]SkinRect* rect)
获取 item 的区域
Rect 返回区域

HRESULT GetGroup([out,retval] IDUITVItemGroup** ppResult)
获取 item 的隶属 group 对象指针
ppResult 返回 group 对象指针

HRESULT GetIndex([out,retval] LONG* nIndex)
获取 item 的序号
nIndex 返回序号

5.10. IDUITVGroup : IDUITVItemBase

HRESULT InsertItem([in]SHORT nIndex,[in]LONG nID,[in]BSTR strText,[out,retval]IDUITVItem** ppResult)
向组中插入 item
nIndex 插入的位置，从 0 开始 nId Item 的 id strText item 的文字 ppResult 返回 item 的对象指针

HRESULT AppendItem([in] LONG nID,[in] BSTR strText,[out,retval]IDUITVItem** ppResult)
向组中增加 item
nId item 的 id strText item 的文字 ppResult 返回 item 对象指针

HRESULT RemoveItem([in]IDUITVItem* pItem,[out,retval]VARIANT_BOOL* pbResult)
删除组中 item
ppItem 需要删除的 item 对象指针 pbResult 返回删除结果

HRESULT RemoveItemByID([in]LONG nID,[out,retval]VARIANT_BOOL* pbResult)
通过 id 号删除 item
nId 需要删除 item 的 id pbResult 返回删除结果

HRESULT RemoveAllItems()
删除所有的 item

HRESULT GetItem([in]LONG nID,[out,retval]IDUITVItem** ppResult)
通过 id 号获取 item
nId item 的 id

ppResult 返回 item 对象指针

HRESULT GetAt([in]SHORT nIndex,[out,retval]IDUITVItemBase** ppResult)
通过序号获取组中的 item
nIndex item 的序号 ppResult 返回 item 对象指针

HRESULT GetCount([out,retval]SHORT* pnResult)
查询组中 item 的数量
pnResult 返回数量

HRESULT SetExpand([in]VARIANT_BOOL bExpand)
是否展开此组
bExpand 是否展开

HRESULT IsExpand([out,retval]VARIANT_BOOL* pbResult)
查询组是否展开
pbResult 返回结果

HRESULT SetHaveGroup([in] VARIANT_BOOL bHaveGroup)
设置是否拥含有有子组
bHaveGroup 返回结果

AddGroup([in]LONG nID,[in]BSTR strName,[out,retval]IDUITVItemGroup** ppResult)
向组中添加一个子组
nId 组的 id strName 组的名称 ppResult 返回组的对象指针

HRESULT RemoveGroupByID([in]LONG nID,[out,retval]VARIANT_BOOL* pbResult)
通过 id 删除子组
nId 组的 id

pbResult 返回删除结果

HRESULT RemoveGroupByName([in]BSTR strName,[out,retval]VARIANT_BOOL* pbResult)
通过名称删除子组
nIndex 返回序号

HRESULT RemoveAllGroup()
删除组中的所有子组

HRESULT GetGroupByName([in]BSTR strName,[out,retval]IDUITVItemGroup** ppResult)
通过名称来获得子组
ppResult 返回组的对象指针

HRESULT GetGroupByID([in]LONG nID,[out,retval]IDUITVItemGroup** ppResult)
通过 id 来获取组指针
ppResult 返回组对象指针

HRESULT GetHaveGroup([out,retval]VARIANT_BOOL* pResult)
查询是否含有子组
pResult 返回子组

HRESULT SetGraphics([in] OLE_HANDLE hBmp,[in] VARIANT_BOOL bRedraw)
设置组的图标
hBmp 图标的句柄 bRedraw 是否重绘

HRESULT GetGraphics([out,retval] OLE_HANDLE* phResult)
获取组的图标句柄
phResult 返回组图标句柄

HRESULT InsertGroup([in]SHORT nIndex,[in]LONG nID,[in]BSTR strName,[out,retval]IDUITVItemGroup**

ppResult)
插入一个子组
nIndex 插入的序号，从 0 开始 nid 组的 id strName 组的名称 ppResult 返回组对象指针

HRESULT SetHaveCloseButton([in]VARIANT_BOOL bHaveButton)
设置组是否含有关闭按钮
bHaveButton 是否有关闭按钮

HRESULT HaveCloseButton([out,retval]VARIANT_BOOL* pbResult)
查询是否含有关闭按钮
pbResult 返回是否含有

HRESULT SetTextColor([in]OLE_COLOR color)
设置组字体颜色
Color 组文字颜色

HRESULT SetIcon([in]IDUIImageBase* pImageBase)
通过 imagebase 设置组的图标
pImageBase 组的图标 imagebase

5.11. IDUITVItem ： IDUITVItemBase

HRESULT SetGraphics([in]DUILVI_STATE eState,[in]OLE_HANDLE hBitmap,[out,retval]VARIANT_BOOL* pbResult)
设置 item 的图标
eState item 状态 hBitmap 图标的图片句柄 pbResult 返回结果

HRESULT GetGraphics([in]DUILVI_STATE eState, [out,retval]OLE_HANDLE* phResult)
查询图标的句柄
eState item 的状态 phRsult 返回图标句柄

HRESULT SetHeight([in] SHORT nHeight,[in] VARIANT_BOOL bRedraw)
设置 item 的独立高度
nHeight 新的高度 bRedraw 是否重绘

HRESULT GetHeight([out,retval] SHORT* pnResult)
查询 item 的高度
pnResult 返回高度

HRESULT Refresh()
刷新 item 节点

HRESULT GetCustomObj([out,retval]IDUIControlBase** ppResult)
自定义模式下，获取 item 的模板对象指针
ppResult 返回模板对象指针

HRESULT GetUnitItem([in]SHORT nCol,[out,retval]IDUIUnitItem** ppResult)
Report 样式下，获取单元格 item
nCol 列序号 ppResult 返回子单元格对象

HRESULT AppendItemText([in]SHORT iSubItem,[in] BSTR szItem,[in] BSTR szURL,[in] VARIANT_BOOL bRedraw)
Report 样式下，添加子单元格文本
iSubItem 列序号 szItem item 文本 szURL itemURL 文本 bRedarw 是否重绘

HRESULT AppendImage([in]SHORT iSubItem,[in] BSTR szImagePath,[in] VARIANT_BOOL bRedraw)
Report 样式下，向单元格中添加图片
iSubItem 列序号 szImagePath 图片的路径 bRedraw 是否重绘

HRESULT SetCheck([in]VARIANT_BOOL bCheck)
开启 Check 模式下，设置 item 是否 check
bCheck 是否 check

5.12. IDUIMenuBar： IDUIControlBase 菜单控件接口说明

HRESULT LoadMenu([in]OLE_HANDLE hMenu,[out,retval]VARIANT_BOOL *pbResult)
通过一个标准菜单句柄 HMENU 来构建一个菜单条
hMenu 一个标准菜单句柄 pbResult 返回设置结果

HRESULT AppendItem([in]OLE_HANDLE hMenu, [in]BSTR strText, [out,retval]VARIANT_BOOL *pbResult)
将一个 HMENU 插入一个 item 的新项
hMenu 项的句柄 HMENU strText 项的文本 pbResult 设置的结果

HRESULT InsertItem([in]OLE_HANDLE hMenu, [in]BSTR strText, [in]SHORT nIndex,[out,retval]VARIANT_BOOL *pbResult)
将一个 HMENU 插入一个新 item 项
hMenu 菜单的句柄 strText 新项的文本 nIndex 插入的位置，从 0 开始 pbResult 返回设置结果

HRESULT DeleteItem([in]SHORT nIndex,[out,retval]VARIANT_BOOL *pbResult)

通过序号删除 Item
nIndex item 的序号，从 0 开始 pbResult 返回设置结果

HRESULT ModifyItem([in]SHORT nIndex, [in]BSTR strText, [out,retval]VARIANT_BOOL *pbResult)
通过序号修改 item 的名称
nIndex item 的序号 strText 新 item 的文本 pbResult 返回设置结果

HRESULT GetMenu([out,retval]OLE_HANDLE* phResult)
获取菜单的句柄
phResult 返回菜单的句柄

HRESULT LoadDUIMenu([in] IDispatch *pDUIMenu,[out,retval]VARIANT_BOOL *pbResult)
通过 DirectUI 的 PopupMenu 来创建菜单
pDUIMenu PopupMenu 菜单对象指针 pbResult 返回设置结果

HRESULT GetMenuItemCount([out, retval]LONG* nCount)
获取菜单项的数量
nCount 返回项数量

5.13. IDUIProgressbar： IDUIControlBase 进度条控件接口说明

HRESULT SetRange([in] SHORT nLower, [in] SHORT nUpper)
设置进度条范围
nLower 最小值 nUpper 最大值

HRESULT GetRange([out] SHORT* pnLower, [out] SHORT* pnUpper)
获取进度条范围

pnLower 返回最小值 pnUpper 返回最大值

HRESULT SetPos([in] SHORT nPos, [out,retval] SHORT* pnResult)
设置进度条显示位置
nPos 位置 pnResult 返回设置结果

HRESULT GetPos([out,retval] SHORT* pnResult)
查询获取当前值
pnResult 返回当前值

HRESULT SetStep([in] SHORT nStep, [out,retval] SHORT* pnResult)
进度条前进 nStep 步，具体一步在皮肤中设置
nStep 前进步数

HRESULT StepIt([out,retval] SHORT* pnResult)
前进一步，具体一步的大小在皮肤中设置
pnResult 返回前进大小

StartAnimate([out,retval] VARIANT_BOOL *pbResult)
Vista 样式下启动动画
pbResult 返回设置结果

StopAnimate([out,retval] VARIANT_BOOL *pbResult)
停止动画
pbResult 返回设置结果

5.14. IDUIScrollbar： IDUIControlBase 滚动条控件接口说明

HRESULT GetScrollPos([out,retval]LONG *pnResult)

获取滚动条位置
pnResult 返回当前滚动位置

GetScrollRange([in,out]LONG* lpMin,[in,out]LONG* lpMax)
查询滚动条滚动范围
lpMin 最小值 lpMax 最大值

HRESULT SetScrollPos([in]LONG nPos,[in,defaultvalue(-1)]VARIANT_BOOL Redraw,[out,retval]LONG *pnResult)
设置滚动条滚动位置
nPos 滚动位置 Redraw 是否重绘 pnResult 返回设置结果

HRESULT SetScrollRange([in]LONG nMin,[in]LONG nMax,[in,defaultvalue(-1)]VARIANT_BOOL bRedraw)
设置滚动范围
nMin 最小值 nMax 最大值 bRedraw 是否重绘

HRESULT EnableScrollBar([in]VARIANT_BOOL bEnabled)
是否启用滚动条
bEnabel 是否启用

HRESULT ShowScrollBar([in,defaultvalue(-1)]VARIANT_BOOL bShow)
是否显示或隐藏滚动条
bShow 是否显示

HRESULT SetHorz([in]VARIANT_BOOL bHorz)
设置滚动条方向是否是水平的
bHorz 是否是水平的

HRESULT IsHorz([out,retval]VARIANT_BOOL *pbResult)
查询滚动条是否是水平的
pbResult 返回结果

HRESULT SetScrollInfo([in]DUISCROLLINFO *lpScrollInfo,[in,defaultvalue(-1)]VARIANT_BOOL bRedraw,[out,retval]VARIANT_BOOL *pbResult)
设置滚动条信息
lpScrollInfo 滚动条信息 bRedraw 是否重绘 pbResult 返回设置结果

HRESULT GetScrollInfo([in]DUISCROLLINFO *lpScrollInfo,[in,defaultvalue(0x001F)]DUISB_MASK sifMask,[out,retval]VARIANT_BOOL *pbResult)
获取滚动信息
lpScrollInfo 返回滚动信息 sifMask 需要查询的内容 pbResult 返回查询是否成功

HRESULT SetOwnerCtrl([in]IDUIControlBase *pOwnerCtrl)
设置滚动条的隶属控件，设置后滚动条将会向该控件发送滚动消息
pOwnderCtrl 隶属控件指针

HRESULT GetOwnerCtrl([out,retval]IDUIControlBase **pResult)
查询隶属的控件
pResult 返回控件指针

HRESULT GetPageSize([out,retval]LONG *pnResult)
查询滚动页数
pnResult 返回页数

HRESULT SetPageSize([in]LONG nPageSize)
设置滚动页数， 已过时，我们使用 SetScrollInfo 来设置

nPageSize

HRESULT Redraw()
滚动条自我重绘

HRESULT SetHorzUpButtonWidth([in]LONG nHorzUpButtonWidth)
设置水平左按钮的宽度
nHorzUpButtonWidht 宽度

HRESULT SetHorzDownButtonWidth([in]LONG nHorzDownButtonWidth)
设置水平右按钮的宽度
nHorzDownButtonWidht 宽度

HRESULT SetVertUpButtonHeight([in]LONG nVertUpButtonHeight)
设置垂直上按钮高度
nVertUpButtonHeight 高度

HRESULT SetVertDownButtonHeight(LONG nVertDownButtonHeight)
设置垂直下按钮高度
nVertDownButtonHeight 高度

5.15. IDUISliderbar： IDUIControlBase 滑动条控件接口说明

HRESULT GetRange([out] LONG* pnMin, [out] LONG* pnMax)
获取滑动条的范围
pnMin 返回最小值 pnMax 返回最大值

HRESULT SetRange([in] LONG nMin, [in] LONG nMax, [in] VARIANT_BOOL bRedraw,[out,retval]VARIANT_BOOL *pbResult)

设置滑动条的最大值
nMin 最小值 nMax 最大值 bRedraw 是否重绘

HRESULT GetRangeMin([out,retval] LONG *pnResult)
获取最小的范围
pnResult 返回最小范围值

HRESULT SetRangeMin([in] LONG nMin, [in] VARIANT_BOOL bRedraw,[out,retval]VARIANT_BOOL *pbResult)
设置最小范围值
nMin 最小范围 bRedraw 是否重绘 pbResult 返回设置结果

HRESULT GetRangeMax([out,retval] LONG *pnResult)
获取最大范围值
pnResult 返回最大范围值

HRESULT SetRangeMax([in] LONG nMax, [in] VARIANT_BOOL bRedraw,[out,retval]VARIANT_BOOL *pbResult)
设置最大范围值
nMax 最大值 bRedraw 是否重绘 pbResult 返回设置结果

HRESULT SetTooltip([in] BSTR strTooltip, [in] VARIANT_BOOL bRedraw,[out,retval]VARIANT_BOOL *pbResult)
设置控件 tooltip
strToolTip tooltip 文本 bRedraw 是否重绘 pbResult 返回设置结果

HRESULT GetTooltip([out,retval] BSTR* pstrResult)

获取 tooltip 文本
pstrResult 返回 tooltip

HRESULT GetPos([out,retval]LONG *pnResult)
当前的位置
pnResult 返回当前位置

HRESULT SetPos([in]LONG nPos, [out,retval]VARIANT_BOOL *pbResult)
设置当前滑动的位置
nPos 滑动的位置 pbResult 返回设置结果

HRESULT SetStatePos([in]LONG nPos, [out,retval]VARIANT_BOOL *pbResult)
设置读取状态的位置（播放器滑块使用）
nPos 位置 pbResult 返回设置结果

HRESULT GetStatePos([out,retval]LONG *pnResult)
获取读取状态的位置（播放器滑块使用）
pnResult 返回位置

5.16. IDUIStatic： IDUIControlBase 文本控件接口说明

HRESULT SetText([in] BSTR strText)
设置文本控件文本
strText 文本内容

HRESULT GetText([out,retval] BSTR* pstrResult)
获取文本内容
pstrResult 返回文本内容

SetTooltip([in] BSTR strTooltip)
设置 tooltip
strTooltip 文本内容

HRESULT GetTooltip([out,retval] BSTR* pstrResult)
获取 tooltip
pstrResult 返回 tooltip 内容

HRESULT StartScroll([out,retval] VARIANT_BOOL* pbResult)
开始滚动
pbResult 返回是否成功

AddScrollItem([in] BSTR strText,[in] BSTR strURL,[out,retval] OLE_HANDLE* phResult)
增加滚动项
strText 文本内容 strURL url 内容 phResult 返回 item 的句柄

HRESULT SetTextScroll([in]VARIANT_BOOL bIsScroll, [in]VARIANT_BOOL bRedraw,[out,retval]VARIANT_BOOL *pbResult)
设置是否可滚动
bIsScroll 是否可滚动 bRedraw 是否重绘 pbResult 返回设置结果

HRESULT GetTextScroll([out,retval]VARIANT_BOOL *pbResult)
查询是否可滚动
pbResult 返回结果

HRESULT SetScrollSpace([in]LONG nSpace,[out,retval]VARIANT_BOOL *pbResult)
设置滚动的间隔
nSpace 间隔大小

pbResult 返回设置结果

HRESULT GetScrollSpace([out,retval]LONG *pnResult)
获取滚动间隔
pnResult 返回间隔

HRESULT SetScrollSpeed([in]LONG nSpeed,[out,retval]VARIANT_BOOL *pbResult)
设置滚动的速度
nSpeed 速度，毫秒 pbResult 返回设置结果

HRESULT GetScrollSpeed([out,retval]LONG *pnResult)
获取滚动速度
pnResult 返回滚动速度，单位毫秒

HRESULT SetScrollStep([in]LONG nStep,[out,retval]VARIANT_BOOL *pbResult)
设置每个时间单位的滚动距离
nStep 滚动距离，单位像素 pbResult 返回设置结果

HRESULT GetScrollStep([out,retval]LONG *pnResult)
获取每个时间单位的滚动距离
pnResult 返回大小

HRESULT DeleteScrollItem([in] OLE_HANDLE hTextItem,[out,retval] VARIANT_BOOL* pbResult)
删除滚动项
hTextItem 滚动项的大小 pbResult 返回结果

HRESULT ModifyScrollItem([in] OLE_HANDLE hTextItem,[in] BSTR strText,[in] BSTR strNewURL,[out,retval] VARIANT_BOOL* pbResult)
修改滚动项的文本内容

hTextItem 滚动项的句柄 strText 文本内容 strNewURL url 内容 pbResult 返回设置结果

HRESULT DeleteAllScrollItem([out,retval] VARIANT_BOOL* pbResult)
删除所有的滚动项
pbResult 返回设置结果

HRESULT GetScrollTextsCount([out,retval] SHORT* pnResult)
返回滚动项的数量
pnResult 返回数量

HRESULT GetScrollText([in] OLE_HANDLE hTextItem,[out,retval] BSTR* pstrResult)
查询滚动项的文本
hTextItem 滚动项的句柄 pstrResult 返回文本内容

HRESULT GetScrollURL([in] OLE_HANDLE hTextItem,[out,retval] BSTR* pstrResult)
查询滚动项 url 文本
hTextitem 滚动项的句柄 pstrResult 文本内容

HRESULT ModifyScrollText([in] OLE_HANDLE hTextItem,[in] BSTR strText,[out,retval] VARIANT_BOOL* pbResult)
修改滚动项的文本
hTextItem 滚动项的句柄 strText 滚动项的文本

HRESULT ModifyScrollURL([in] OLE_HANDLE hTextItem,[in] BSTR strNewURL,[out,retval] VARIANT_BOOL* pbResult)
修改滚动项的 url 链接地址
hTextItem 滚动项句柄 strNewURL url 地址

pbResult 返回设置结果

HRESULT SetImageByPath([in] BSTR strImagePath,[in] SHORT nFrames,[in] VARIANT_BOOL bHorz)
通过图片路径设置背景
strImagePath 图片路径 nFrames 保留，设置为 1 bHorz 保留，设置为 VARIANT_TRUE

HRESULT SetImageBack([in] OLE_HANDLE hBmp, [in] DUI_STATICSTATE eState)
通过一个 HBitmap 设置背景
hBmp 图片句柄 eState 状态

HRESULT SetBackImageSec([in] DUI_STATICSTATE eState,[in]IDUIImageBase* pImageBase)
通过 imagebase 来设置背景
eState 状态 pImageBase 图片 imagebase 指针

HRESULT PauseScroll([in] VARIANT_BOOL bPause,[out,retval] VARIANT_BOOL* pbResult)
是否暂停滚动
bPause 是否暂停 pbResult 返回设置结果

5.17. IDUITabCtrl： IDUIControlBase 标签页控件接口说明

HRESULT AppendItem([in]BSTR strName,[in]LONG nID,[in]BSTR strText,[out,retval]IDUITabCtrlItem **pResult)
插入一个 item
strName item 的名称，不能重名 nId item 的 id，不能重名 strText item 的显示文本 pResult 返回 item 对象指针

HRESULT InsertItem([in]SHORT nIndex,[in]BSTR strName,[in]LONG nID,[in] BSTR strText,[out,retval]IDUITabCtrlItem **pResult)
插入一个 item
nIndex 插入的位置，从 0 开始 strName item 的名称，不能重名 nId item 的 id，不能重名 strText item 的显示文本 pResult 返回 item 对象指针

HRESULT GetItemByIndex([in]SHORT nIndex,[out,retval]IDUITabCtrlItem **pResult)
通过索引来获取 item 对象指针
nIndex 索引的序号，从 0 开始 pResult 返回 item 对象指针

HRESULT GetItemByName([in]BSTR strName,[out,retval]IDUITabCtrlItem **pResult)
通过名称来获取 item 对象指针
strName item 的名称 pResult item 对象指针

HRESULT GetItemByID([in] LONG nID, [out,retval] IDUITabCtrlItem **pResult)
通过 Id 来获取 item 对象指针
nId item 的 id pResult 返回 item 对象指针

HRESULT GetItemCount([out,retval] LONG* plResult)
获取 item 的数量
plResult 返回数量

HRESULT RemoveItem([in] IDUITabCtrlItem *pItem, [out,retval] VARIANT_BOOL* pbResult)
删除某一个 item
pItem 需要删除的 item 对象指针 pbResult 返回删除结果

HRESULT RemoveItemByIndex([in] SHORT nIndex, [out,retval] VARIANT_BOOL* pbResult)

通过序号删除
nIndex item 的序号，从 0 开始 pResult 返回删除结果

HRESULT RemoveItemByName([in] BSTR strName, [out,retval] VARIANT_BOOL* pbResult)
通过名称删除
strName item 的名称 pbResult 返回设置结果

HRESULT RemoveItemByID([in] LONG nID,[out,retval]VARIANT_BOOL* pbResult)
通过 id 删除某一个 item
nId item 的 id pbResult 返回删除结果

HRESULT RemoveAllItems()
删除所有的 item

HRESULT SwapItem([in]IDUITabCtrlItem* pFirst,[in]IDUITabCtrlItem* pSecond)
交换两个 item
pFirst 第一个 item 对象指针 pSecond 第二个 item 对象指针

HRESULT SetDragable([in]VARIANT_BOOL bDragable)
设置 tab 的 item 是否可拖拽来交换位置
bDragable 是否可拖拽

HRESULT EnableItem([in] LONG nIdIndex, [in] VARIANT_BOOL bEnable, [in] VARIANT_BOOL bByIndex)
通过序号或者 id 来禁用某一个 item
nIdIndex item 的序号或者 id bEnable 是否启用 bByIndex 是否为序号，否则为 id

HRESULT IsEnableItem([in] LONG nIdIndex, [in] VARIANT_BOOL bByIndex, [out, retval]VARIANT_BOOL* pEnable)
查询某一个 item 是否禁用
nIdIndex item 的序号或者 id bByIndex 是否为序号，否则为 id pEnable 返回结果

HRESULT SetControlID([in] SHORT nIndex)
设置控件的 id
nIndex 控件 id

HRESULT GetControlID([out,retval] SHORT * pnResult)
查询控件的 id
返回 id

5.18. IDUITabCtrlItem ： IDispatch

HRESULT SetName([in] BSTR strName)
设置 item 的名称
strName 名称文本

HRESULT GetName([out,retval] BSTR* strName)
获取 item 的名称
strName item 名称

HRESULT SetText([in] BSTR strText)
设置 item 的显示文本
strText 显示文本

HRESULT GetText([out,retval] BSTR* pResult)
查询 item 的显示文本
pResult 返回显示文本

HRESULT SetID([in] LONG nID)
设置 item 的 id
Id item 的 id

HRESULT GetID([out,retval]LONG* plResult)
获取 Item 的 id
plResult 返回 id

HRESULT SetToolTip([in] BSTR strTipText)
设置 item 的 tooltip 文本
strTipText tooltip 文本

HRESULT GetToolTip([out,retval] BSTR* psResult)
获取 Item 的 tooltip 文本
psResult 返回 tooltip 文本

HRESULT SetVisible([in] VARIANT_BOOL bVisible)
设置 item 是否可见
bVisible 是否可见

HRESULT IsVisible([out,retval] VARIANT_BOOL* pbResult)
查询 item 是否可见
pbResult 返回是否可见

HRESULT GetRect([out,retval] SkinRect* pResult)
查询 item 的窗口上的区域
pResult 返回区域

HRESULT SetGraphic([in] OLE_HANDLE hBitmap)

通过 hbitmap 设置 item 的图标
hBitmap 图片句柄

HRESULT GetGraphic([out,retval] OLE_HANDLE* phResult)
查询 item 的图标句柄
phResult 返回图标句柄

HRESULT SetIconFilePath([in] BSTR sFilePath)
通过路径设置 item 的图标
sFilePath 图标路径

HRESULT GetIconFilePath([out,retval] BSTR* sFilePath)
查询 item 的图标路径
sFilePath 返回路径

HRESULT GetIndex([out, retval]LONG* nID)
查询 item 的序号
nId 返回序号

HRESULT SetHaveCloseButton([in]VARIANT_BOOL bHaveButton,[in] VARIANT_BOOL bRedraw)
设置 item 拥有关闭按钮
bHaveButton 是否拥有关闭按钮 bRedraw 是否重绘

HRESULT IsHaveCloseButton([out,retval]VARIANT_BOOL* pbResult)
查询 item 是否拥有关闭按钮
pbResult 返回结果

HRESULT SetWidth([in]LONG nWidth,[in]VARIANT_BOOL bReCalc)
设置 item 的宽度
nWidht 返回宽度 bReCalc 返回是否重新计算

HRESULT GetWidth([out,retval] LONG* plResult)
获取 item 的宽度
plResult 返回宽度

5.19. IDUIToolBar： IDUIControlBase 工具栏控件接口说明

HRESULT InsertItem([in]SHORT nIndex,[in] LONG nID,[in] BSTR strText,[in]LONG nGroupID,[in] DUI_TBITEM_STYLE eStyle,[in]BSTR strToolTip, [out,retval] IDUIToolBarItemBase** ppResult)
插入一个 toolbar 项
nIndex 插入的位置，从 0 开始 nid 项的 id strText 项的文本 nGroupID 如果为 Radio，则 group 的 id eStyle 项的类型 strToolTip tooltip 文本 返回 item 的对象指针

HRESULT AppendItem([in] LONG nID,[in] BSTR strText,[in]LONG nGroupID,[in] DUI_TBITEM_STYLE eStyle,[in]BSTR strToolTip,[out,retval] IDUIToolBarItemBase** ppResult)
增加一个 toolbar 项
nid 项的 id strText 项的文本 nGroupID 如果为 Radio，则 group 的 id eStyle 项的类型 strToolTip tooltip 文本 返回 item 的对象指针

HRESULT RemoveItemByID([in]LONG nID,[out,retval] VARIANT_BOOL *pbResult)
通过 item 的 id 来删除 item
nId item 的 id pbResult 返回删除的结果

HRESULT RemoveItemByIndex([in]SHORT nIndex,[out,retval] VARIANT_BOOL *pbResult)
通过序号来删除 item

nIndex item 的序号，从 0 开始 pbResult 返回删除的结果

HRESULT RemoveAllItems()
删除所有的 item

HRESULT GetItem([in]LONG nID,[out,retval] IDUIToolBarItemBase** ppResult)
通过 id 来获取 item 对象指针
nId item 的 id ppResult 返回 item 对象指针

HRESULT GetAt([in]SHORT nIndex,[out,retval] IDUIToolBarItemBase** ppResult)
通过序号来获取 item 对象指针
nIndex 序号 ppResult 返回对象指针

HRESULT GetItemCount([out,retval] LONG *pnResult)
查询 item 的数量
pnResult 返回数量

HRESULT CancelGroupRadio([in] LONG nGnGroupID,[out,retval] VARIANT_BOOL* pbResult)
取消 Radio 的选择
nGnGroupId 组的 id pbResult 返回设置结果

HRESULT SetShowText([in]VARIANT_BOOL bShowText,[in]VARIANT_BOOL bRedraw)
设置是否显示 item 下文字
bShowText 是否显示 bRedraw 是否重绘

HRESULT IsShowText([out,retval]VARIANT_BOOL* pbResult)
查询是否显示文字

pbResult　返回结果

5.20. IDUIStarCtrl： IDUIControlBase 星型控件接口说明

HRESULT GetFrames([out,retval] SHORT *pnResult)
查询可显示的最大数量
pnResult 返回数量

HRESULT SetFrames([in] SHORT nFrames, [in] VARIANT_BOOL bRedraw,[out,retval]VARIANT_BOOL *pbResult)
设置可显示最大数量
nFrames 数量 bRedraw 是否重绘 pbResult 返回结果

HRESULT GetCurrPos([out,retval] SHORT *pnResult)
得到当前的位置
pnResult 返回位置

HRESULT SetCurrPos([in] SHORT nPos, [in] VARIANT_BOOL bRedraw,[out,retval]VARIANT_BOOL *pbResult)
设置当前的位置
nPos 位置 bRedraw 是否重绘 pbResult 返回是否设置成功

HRESULT IsCanSelStar([out,retval]VARIANT_BOOL* bCanSelStar)
是否可以通过鼠标选择数量
bCanSelStar 返回结果

HRESULT SetCanSelStar([in]VARIANT_BOOL bCanSelStar)
设置当前是否可通过鼠标点击选择数量

bCanSelStart 是否可选择

5.21. IDUIComboBox： IDUIControlBase 下拉控件接口说明

HRESULT SetCurrText([in]BSTR strText)
设置当前显示文本
strText 文本内容

HRESULT GetCurrText([out,retval]BSTR *pResult)
获取当前文本
pResult 文本内容

HRESULT SetDefText([in]BSTR strText)
设置默认文本
strText 默认文本

HRESULT GetDefText([out,retval]BSTR *pResult)
获取默认文本
pResult 返回默认文本

HRESULT AddItem([in]BSTR strText,[in]LONG nID,[in]OLE_HANDLE hInfo,[out,retval]LONG *pnResult)
增加项
strText 项的文本 nId 项的 id hInfo 项的一个附加数据，可为 NULL pnResult 返回位置

HRESULT SetItemText([in]LONG nIndex,[in]BSTR strText,[out,retval]VARIANT_BOOL *pbResult)
设置 item 的文本
nIndex item 的序号 strText 文本内容 pbResult 返回设置结果

HRESULT GetItemText([in]LONG nIndex,[out,retval]BSTR *pResult)
通过序号查询 item 的文本
nIndex 序号 pResult 返回文本

HRESULT RemoveAt([in]LONG nIndex,[out,retval]VARIANT_BOOL *pbResult)
通过序号删除 item
nIndex 序号 pbResult 返回删除结果

HRESULT RemoveAll([out,retval]VARIANT_BOOL *pbResult)
删除所有的 item
pbResult 返回删除结果

HRESULT GetItemCount([out,retval]LONG *pnResult)
获取 item 的刷另
pnResult 返回数量

HRESULT SetCurSel([in]LONG nSelect)
设置当前选择 item
nSelect 序号,从 0 开始

HRESULT GetCurSel([out,retval]LONG *plResult)
获得当前选择的 item 序号
plResult 返回当前选择 item 序号

HRESULT SetPopupList([in]IDispatch *pPoputList)
设置此控件绑定的 IDUIPopupSingleList
pPoputList 绑定的控件指针

HRESULT SetMaxItemCount([in]LONG nItemCount)
可显示的 item 的最大个数
nItemCount 最大个数

HRESULT GetMaxItemCount([out,retval]LONG *nItemCount)
查询 item 的数量
nItemCount item 的最大数量

HRESULT SetData([in] SHORT nIndex, [in] OLE_HANDLE hData)
给 Item 设置一个额外数据
nIndex item 的序号，从 0 开始 hData 额外的数据

HRESULT GetData([in] SHORT nIndex, [out,retval] OLE_HANDLE* phData)
通过序号为 item 设置一个额外数据
nIndex 序号 phData 额外数据

5.22. IDUIPopupMenu： IDUIControlBase 弹出菜单控件接口说明

HRESULT TrackPopupMenu([in] DUI_TPMSTYLE eFlags,[in] SHORT nX,[in] SHORT nY,[in] OLE_HANDLE hWnd)
弹出菜单
eFalgs 弹出位置，保留，设置为 DUI_TPM_TOPALIGN nX 弹出的水平位置 nY 弹出的垂直位置 hWnd 接受消息的窗口

HRESULT GetMenu([in] LONG nItem,[in] VARIANT_BOOL bByPos,[in] DUI_MENUITEM_STYLE eStyle,[out,retval]IDUIMenuItemBase** ppResult)
获取菜单中的 item
nItem item 的 id 或者序号 bByPos 是否通过序号查询

eStyle item 的类型 ppResult 返回 item 对象指针

HRESULT GetMenuItemCount([out,retval] LONG* pnResult)
查询 item 的数量
pnResult 返回数量

HRESULT GetSubMenu([in] SHORT nIndex,[out,retval] IPopMenu** ppResult)
获取子菜单
nIndex 序号 ppResult 返回子菜单对象指针

HRESULT EnableMenuItem([in]LONG nItem,[in]VARIANT_BOOL bByPos,[in]VARIANT_BOOL bEnable)
启用某一个 item
nItem item 的序号或者 id bByPos 是否通过序号来查询 bEnable 是否启用，否则禁用

HRESULT SetEventRecieve([in]IDirectUI* pDirectUI)
设置菜单 DUIMessage 的发送的 directui 对象
pDirectUI directui 对象指针

HRESULT GetEventRecieve([out,retval]IDirectUI** ppResult)
查询 DUIMessage 接受的 directui 对象
ppResult 返回 directui 对象指针

HRESULT SetAutoCheck([in]VARIANT_BOOL bAutoCheck)
设置 check item 是否自动 check
bAutoCheck 是否自动 check，否则程序调用 check 方法 check item

HRESULT GetAutoCheck([out,retval] VARIANT_BOOL* pbResult)
获取是否开启自动 check

pbResult 返回结果

HRESULT TrackPopupMenuEx([in] DUI_TPMSTYLE eFlags,[in] SHORT nX,[in] SHORT nY,[in] OLE_HANDLE hWnd,[out,retval]LONG* plResult)
弹出菜单，调用等待菜单选择后再继续执行后面的代码，类 TPM_RETURNCMD 样式
eFalgs 弹出位置，保留，设置为 DUI_TPM_TOPALIGN nX 弹出的水平位置 nY 弹出的垂直位置 hWnd 接受消息的窗口

HRESULT SetUseStandardMenu([in]VARIANT_BOOL bAutoCheck)
是否使用标准弹出菜单
bAutoCheck 是否使用标准菜单

HRESULT GetUseStandardMenu([out,retval] VARIANT_BOOL* pbResult)
查询是否使用标准弹出菜单
pbResult 返回结果

HRESULT SetStandardMenuHanlde([in]OLE_HANDLE hMenuHandle)
通过 HMENU 来构建菜单
hMenuHandle 传入 HMENU

HRESULT GetStandardMenuHanlde([out,retval]OLE_HANDLE* hMenuHandle)
获取构建的 HMENU
hMenuHandle 返回 HMENU

5.23. IPopMenu : IDispatch

HRESULT TrackPopupMenu([in] DUI_TPMSTYLE eFlags,[in] SHORT nX,[in] SHORT nY,[in] OLE_HANDLE hWnd)
弹出菜单
eFlags 弹出选项，固定设置为 TOPALIGN

nX 弹出水平位置 nY 弹出垂直位置 hWnd 弹出窗口的父窗口

HRESULT AppendMenu([in] LONG nID,[in] BSTR strName,[in] BSTR strText,[in] LONG nGroupID,[in] DUI_MENUITEM_STYLE eStyle,[in]VARIANT_BOOL bIsPopupStyle, [in]VARIANT_BOOL bRecalculate, [out,retval] IDUIMenuItemBase** ppResult)
添加 item
nId item 的 id strName item 的名称 strText item 显示的文本 nGroupId 如果为 radio 样式，则此 item 的所属 group id eStyle item 的样式 bIsPopupStyle 是否是拥有子菜单 bRecalculate 是否重新计算位置 ppResult 返回 item 对象指针

HRESULT InsertMenu([in] SHORT nIndex,[in] LONG nID,[in] BSTR strName,[in] BSTR strText,[in] LONG nGroupID,[in] DUI_MENUITEM_STYLE eStyle,[in]VARIANT_BOOL bIsPopupStyle, [in]VARIANT_BOOL bRecalculate,[out,retval] IDUIMenuItemBase** ppResult)
插入一个 item
nIndex 插入的位置，从 0 开始 nId item 的 id strName item 的名称 strText item 显示的文本 nGroupId 如果为 radio 样式，则此 item 的所属 group id eStyle item 的样式 bIsPopupStyle 是否是拥有子菜单 bRecalculate 是否重新计算位置 ppResult 返回 item 对象指针

HRESULT RemoveMenu([in] LONG nItem,[in]VARIANT_BOOL bByPos,[out,retval] VARIANT_BOOL* pbResult)
删除 item
nItem item 的序号或者 id bByPos 是否通过序号查找否则为 id pbResult 返回删除结果

HRESULT GetMenu([in] LONG nItem,[in] VARIANT_BOOL bByPos,[in] DUI_MENUITEM_STYLE

eStyle,[out,retval]IDUIMenuItemBase** ppResult)
获取 item 对象指针
nItem item 的序号或者 id bByPos 是否通过序号查找否则为 id eStyle item 的类型 ppResult 返回 item 对象指针

HRESULT GetMenuItemCount([out,retval] LONG* pnResult)
获取 item 的数量
pnResult 返回刷另

HRESULT GetSubMenu([in] SHORT nIndex,[out,retval] IPopMenu** ppResult)
获取子菜单
nIndex 序号 ppResult 返回子菜单对象指针

HRESULT IsMenuVisible([out,retval] VARIANT_BOOL *pbResult)
菜单是否可见
pbResult 是否可见

HRESULT GetStandardMenuHanlde([out,retval]OLE_HANDLE* hMenuHandle)
通过 hMenuHandle 来构建菜单，设置构建源 HMENu
hMenuHandle 传入 HMENU

HRESULT GetHandle([out,retval] OLE_HANDLE* phResult)
获取构建的 HMENU
hMenuHandle 返回 HMENU

HRESULT SetMenuPos([in] SHORT x,[in] SHORT y)
设置菜单的位置
X 水平位置 Y 垂直位置

HRESULT GetMenuTextByIndex([in] SHORT nIndex,[out,retval] BSTR* pstrResult)
通过序号来获取 item 的文本
nIndex 序号 pstrResult 返回 item 文本

HRESULT GetMenuItemID([in] SHORT nIndex,[out,retval] LONG* plResult)
通过序号来获取 item 的 id
nIndex 序号 plResult 返回 id

HRESULT SetItemVisible([in]LONG nItem,[in]VARIANT_BOOL bByPos,[in]VARIANT_BOOL bVisible)
设置 item 是否可见
nItem item 的序号或者 id bByPos 是否通过序号查找否则为 id bVisible 是否可见

HRESULT EnableMenuItem([in]LONG nItem,[in]VARIANT_BOOL bByPos,[in]VARIANT_BOOL bEnable)
是否启用或禁用某个 item
nItem item 的序号或者 id bByPos 是否通过序号查找否则为 id bEnable 是否启用，否则禁用

RESULT UnSelRadioGroup([in]LONG nMenuBar)
是否取消 radio 选中
nMenubar radio 的组 id

5.24. IDUIMenuItemBase ： IDispatch

HRESULT SetID([in] LONG nID)
设置 item 的 id
nId item 的 id

HRESULT GetID([out,retval] LONG* plResult)
获取 item 的 id
plResult 返回 id

HRESULT GetStyle([out,retval] DUI_MENUITEM_STYLE *peResult)
查询 item 的类型
peResult 返回样式

HRESULT SetVisible(VARIANT_BOOL bVisible)
设置是否可见
bVisible 是否可见

HRESULT GetVisible(VARIANT_BOOL* pbResult)
查询是否可见
pbResult 返回是否可见

5.25. IDUIMenuItem ： IDUIMenuItemBase

HRESULT SetText(BSTR strText)
设置显示文本
strText 显示文本

HRESULT GetText(BSTR* pstrResult)
获取显示文本
pstrResult 返回显示文本

HRESULT SetData(OLE_HANDLE pData)
设置 item 的额外数据
pData item 的数据

HRESULT GetData(OLE_HANDLE* phResult)
获取 item 的额外数据
phResult 返回额外数据

HRESULT SetHotKey(BSTR strHotKey)
设置 item 的快捷键
strHotKey 快捷键文本

HRESULT GetHotKey(BSTR* pstrResult)
获取快捷键
pstrResult 返回快捷键文本

5.26. IDUIMenuPushItem ： IDUIMenuItem

HRESULT SetGraphic([in] BSTR strImage)
通过路径设置 item 的图标
strImage 图标的路径

HRESULT GetGraphic([out,retval] BSTR* pstrResult)
获取图标路径
pstrResult 返回图标路径

HRESULT SetPopStyle([in]VARIANT_BOOL bIsPopupStyle)
设置是否拥有子菜单风格
bIsPopupStyle 是否拥有子菜单风格

HRESULT GetPopStyle([out,retval]VARIANT_BOOL* bIsPopupStyle)
查询是否拥有子菜单风格
bIsPopupStyle 是否有子菜单风格

5.27. IDUIMenuCheckItem : IDUIMenuItem

HRESULT SetCheck([in] DUI_MENUITEM_VALUE eValue)
设置是否 check
eValue 是否 check

HRESULT GetCheck([out,retval] DUI_MENUITEM_VALUE* peResult)
查询是否已经 check
peResult 返回结果

5.28. IDUIMenuRadioItem : IDUIMenuItem

HRESULT SetGroupID([in] LONG nID)
设置组的 id
nId 组的 id

HRESULT GetGroupID([out,retval] LONG* plResult)
获取组的 id
plResult 返回组 id

HRESULT SetRadio()
选中该 item

HRESULT GetValue([out,retval] DUI_MENUITEM_VALUE* peResult)

HRESULT GetCheck([out,retval] DUI_MENUITEM_VALUE* peResult)
查询是否选中
peResult 返回是否选中

5.29. IDUILogoObj： IDUIControlBase 头像控件接口说明

HRESULT SetLogoImage([in] BSTR strFileName, [out,retval] VARIANT_BOOL* pbResult)
通过路径来设置 logo 图片
strFileName 图片路径 pbResult 返回设置结果

HRESULT DestroyLogoImage()
效果 logo 缓存

HRESULT SetImageHandle([in] OLE_HANDLE hBitmap)
通过 HBITMAP 来设置 logo
hBitmap logo 的句柄

HRESULT SetBackImage([in] IDUIImageBase* pImageBase,[in] VARIANT_BOOL bRedraw,[out,retval] VARIANT_BOOL* pbResult)
通过 imagebase 来设置背景
pImageBase 通过 imagebase 来设置背景 bRedraw 是否重绘 pbResult 返回设置结果

HRESULT GetBackImage([out,retval] IDUIImageBase** ppImageBase);
获取背景 imagebase
ppImageBase 返回背景 imagebase

5.30. IDUIFormBorder： IDUIControlBase 边框控件接口说明

HRESULT SetCaption([in] BSTR bstrCaption,[in] VARIANT_BOOL bRefresh)
设置标题
bstrCaption 标题文本 bRefresh 是否重绘

HRESULT SetIcon([in] OLE_HANDLE hBitmapIcon,[in] VARIANT_BOOL bRefresh, [in]VARIANT_BOOL bIsIcon)

通过 HBitmap 或者 HIcon 来设置图标
hBitmapIcon 图标 Hbitmap 或者 Hicon bRefresh 是否重绘 bIsIcon 传入的是否是 HICON

HRESULT SetIconByImageBase([in] IDUIImageBase* hImageBase,[in] VARIANT_BOOL bRefresh)
通过 imagebase 来设置 imagebase
hImageBase 图标 imagebase bRefresh 是否重绘

HRESULT SetCaptionHeight([in] LONG nHeight)
设置标题的高度
nHeight 标题的高度

HRESULT SetUseSysMenu([in] VARIANT_BOOL bUseSysMenu)
点击图标是否弹出标准菜单
bUseSysMenu

HRESULT SetSysBtn([in]BSTR strCloseBtn, [in]BSTR strMaxBtn, [in]BSTR strMinBtn, [in]BSTR strHelpBtn)
设置系统按钮对应的 directui 控件名称
strCloseBtn 关闭按钮 strMaxBtn 最大化按钮 strMinBtn 最小化按钮 strHelpBtn 帮助按钮

HRESULT SetCaptionDragable([in]VARIANT_BOOL bValue)
设置是否通过标题来拖拽窗口
bValue 是否可拖拽

HRESULT DBClickCaptionMaxWnd([in]VARIANT_BOOL bValue)
双击标题栏是否可最大化窗口
bValue 是否双击最大化

5.31. IDUICanlendar： IDUIControlBase 日历控件接口说明

HRESULT CancelEventDate(DUI_CALENDARDATE date, VARIANT_BOOL bRefresh)
取消事件日期
Data 日期 bRefresh 是否重绘

HRESULT SetEventDate(DUI_CALENDARDATE date, VARIANT_BOOL bRefresh)
设置时间日期
Data 日期 bRefrsh 是否重绘

HRESULT IsEventDate(DUI_CALENDARDATE date, VARIANT_BOOL* bpIsEventDay)
查询某一天是否设置了事件时间
Date 日期 bpIsEventDay 返回是否是时间日期

HRESULT JumpToToday()
跳转到今天

HRESULT JumpToSpecDay(SHORT year, SHORT month, SHORT day)
跳转到特定的日期
Year 年 Month 月 Day 日

6. OfficeControls 控件接口介绍

4.1 IDUIOutLookBar： IDUIControlBase OutLookbar 控件接口说明

HRESULT AddFolder([in] BSTR strText, [in] LONG exData,[in] VARIANT_BOOL bRedraw,[out,retval]OLE_HANDLE* phResult)

增加一个组
strText 组的名称 exData 额外的数据 bRedraw 是否重绘 phResult 返回组的句柄

HRESULT RemoveAllItems([in] OLE_HANDLE hFolder,[out,retval] VARIANT_BOOL* pbResult)
删除组中所有的 item
hFolder 组的句柄 pbResult 返回删除结果

HRESULT RemoveAllFolders([out,retval] VARIANT_BOOL* pbResult)
删除所有的组
pbResult 返回删除结果

HRESULT SetItemData([in] OLE_HANDLE hFolder,[in] OLE_HANDLE hItem,[in] LONG dwData)
设置 Item 的额外数据
hFolder 组的句柄 hItem item 的句柄 dwData 额外的数据

HRESULT GetItemData([in] OLE_HANDLE hFolder,[in] OLE_HANDLE hItem,[out,retval] LONG* pdwData)
查询 item 的额外数据
hFolder 组的句柄 hItem item 的句柄 pdwData 返回数据

HRESULT GetItemCount([in]OLE_HANDLE hFolder,[out,retval] SHORT* pnResult)
获取组中 item 的数量
hFolder 组的句柄 pnResult 返回数量

HRESULT RemoveItem([in]OLE_HANDLE hFolder,[in] OLE_HANDLE hItem,[in] VARIANT_BOOL bRedraw,[out,retval] VARIANT_BOOL* pbResult)

删除 item
hFolder 组的句柄 hItem item 的句柄 bRedraw 是否重绘 pbResult 返回删除结果

HRESULT SetItemText([in]OLE_HANDLE hFolder,[in] OLE_HANDLE hItem,[in] BSTR text,[out,retval]VARIANT_BOOL* pbResult)
设置 item 的文本
hFolder 组的句柄 hitem item 的句柄 text item 的显示文本 pbResult

HRESULT GetItemText([in]OLE_HANDLE hFolder,[in] OLE_HANDLE hItem,[out,retval] BSTR* pstrResult)
获取 item 的文本
hFolder 组的句柄 hItem item 的句柄 pstrResult 返回文本

HRESULT GetSelectedItem(OLE_HANDLE* phResult)
查询当前选择到的 item
phResult 返回 item 句柄

HRESULT SetSelectedItem(OLE_HANDLE hItem)
设置选择的 item
hItem 需要选择 item 的句柄

4.2 IDUISimpleTree： IDUIControlBase 树形控件接口说明

遍历树的示例代码

```
OLE_HANDLE hRoot = m_pTreeNote->GetRootItem(NULL); // 得到根节点
OLE_HANDLE hChild = m_pTreeNote->GetChildItem(hRoot); // 遍历所有的子节点
while (hChild)
{
```

```
    TravelTree(m_pTreeNote, hChild); // 遍历子字节
    hChild = m_pTreeNote->GetNextSiblingItem(hChild); // 下一个节点
}
void TravelTree (IDUISimpleTree* pTree, OLE_HANDLE hItem)
{
    CString strText = pTree->GetItemText(hItem).c_str(); // 得到文本
    int n = pTree->GetChildItemCount(hItem); // 子节点数量
    if (n > 0)
    {
        OLE_HANDLE hChild = pTree->GetChildItem(hItem); // 子节点
        while (hChild)
        {
            TrevalTree(pTree, hChild);
            hChild = pTree->GetNextSiblingItem(hChild); // 下一个子节点
        }
    }
}
```

HRESULT InsertChild([in] OLE_HANDLE hParent,[in] OLE_HANDLE hInsertAfter,[in] BSTR strText,[in] VARIANT_BOOL bRedraw,[out,retval] OLE_HANDLE* phResult)
插入节点
hParent 父节点句柄，如果是 NULL 则插入到根节点 hInsertAfter 插入节点的句柄，如果为 NULL 则插入到末尾 strText 节点文本 bRedraw 是否重绘 phResult 返回插入的节点句柄

HRESULT GetChildItem([in] OLE_HANDLE hParent,[out,retval] OLE_HANDLE* phResult)
获取第一个子节点
hParent 父节点句柄，如果为 NULL 则返回为根节点句柄 phResult 返回句柄

HRESULT ItemHasChildren([in]OLE_HANDLE hParent,[out,retval]VARIANT_BOOL* pbResult)
查询节点是否有子节点
hParent 需要查询的节点句柄 pbResult 返回结果

HRESULT GetParentItem([in] OLE_HANDLE hItem,[out,retval] OLE_HANDLE* phResult)

获取节点的父节点
hItem 需要查询的节点 phResult 返回节点句柄

HRESULT GetRootItem([in] OLE_HANDLE hItem,[out,retval] OLE_HANDLE* phResult)
获取节点的根节点
hItem 需要查询的节点句柄 phResult 返回结果句柄

HRESULT DeleteItem([in] OLE_HANDLE hItem,[in] VARIANT_BOOL bRedraw,[out,retval] VARIANT_BOOL* pbResult)
删除节点
hItem 需要删除的节点句柄 bRedraw 是否重绘 pbResult 返回删除是否成功

. HRESULT Expand([in] OLE_HANDLE hItem,[in] VARIANT_BOOL bRedraw,[out,retval] VARIANT_BOOL* pbResult)
展开某一个 item
hItem 需要展开 item 的句柄 bRedraw 是否重绘 pbResult 返回设置结果

HRESULT SelectItem([in] OLE_HANDLE hItem,[out,retval] VARIANT_BOOL* pbResult)
选择某一个控件
hItem 选择控件的句柄 pbResult 返回设置结果

HRESULT SetItemText([in] OLE_HANDLE hItem,[in]BSTR strText,[in] VARIANT_BOOL bRedraw,[out,retval] VARIANT_BOOL* pbResult)
设置 item 的文本
hItem 需要设置的 item 的句柄 strText 文本内容 bRedraw 是否重绘 pbResult 返回设置结果

HRESULT GetItemText([in] OLE_HANDLE hItem,[out,retval] BSTR* pstrResult)
获取节点文本
hItem 节点句柄 pstrResult 返回文本

HRESULT AppendChild([in] OLE_HANDLE hParent,[in] BSTR strText,[in] VARIANT_BOOL bRedraw,[out,retval] OLE_HANDLE* phResult)
增加一个节点到父节点末尾
hParent 父节点句柄，如果是 NULL 则添加到根节点 strText 节点文本 bRedraw 是否重绘 phResult 返回插入的节点句柄

HRESULT SetItemIconByPath([in] OLE_HANDLE hItem,[in] BSTR strPath,[in] BSTR strSelPath,[in] OLE_COLOR clrTrans,[out,retval] VARIANT_BOOL *pbResult)
通过路径来设置一个 item 的图标
hItem item 的句柄 strPath 图标的路径 strSelPath 选择状态下图标的路径 clrTrans 透明色 pbResult 返回设置结果

HRESULT SetItemData([in]OLE_HANDLE hItem,[in]OLE_HANDLE hData,[out,retval]VARIANT_BOOL* pbResult)
给 item 设置一个附加数据
hItem item 的句柄 hData 附加数据 pbResult 返回设置结果

HRESULT GetItemData([in]OLE_HANDLE hItem,[out,retval]OLE_HANDLE *phData)
获取 item 的附加数据
hItem item 的句柄 phData 附加的数据

HRESULT GetNextSiblingItem([in] OLE_HANDLE hItem,[out,retval] OLE_HANDLE* phResult)
获取下一个节点
hItem 前一个节点 phResult 返回前一个节点的下一个节点

HRESULT GetPrevSiblingItem([in] OLE_HANDLE hItem,[out,retval] OLE_HANDLE* phResult)
获取上一个节点
hItem 前一个节点 phResult 返回前一个节点的上一个节点

HRESULT GetItemCount([out,retval] SHORT* pnResult)
查询 item 的数量
pnResult 返回数量

HRESULT GetSelectedItemCount([in] SHORT* pnResult)
获取已经选择 item 的数量
pnResult 返回数量

HRESULT GetFirstSelectedItem([out,retval] IDUITreeItem** ppResult)
获取第一个选择 item
ppResult 返回 item 指针

HRESULT GetNextSelectedItem([in] IDUITreeItem* pItem,[out,retval] IDUITreeItem** ppResult)
获取下一个 item 指针
pItem 前一个 item 指针 ppResult 返回下一个 item 指针

HRESULT GetCheckItemCount([out,retval] SHORT* pnResult)
获取 checkitem 的数量
pnResult 返回 checkitem 的数量

HRESULT GetFirstCheckItem([out,retval] IDUITreeItem** ppResult)

获取第一个 check 的 item
ppResult 返回第一个 checkitem 节点指针

HRESULT GetNextCheckItem([in] IDUITreeItem* pItem,[out,retval] IDUITreeItem** ppResult)
获取下一个 check item 的节点
pItem 前一个 item ppResult 返回下一个节点指针

HRESULT SetScrollTop()
滚动到最上面

HRESULT SetScrollBottom()
滚动到最底部

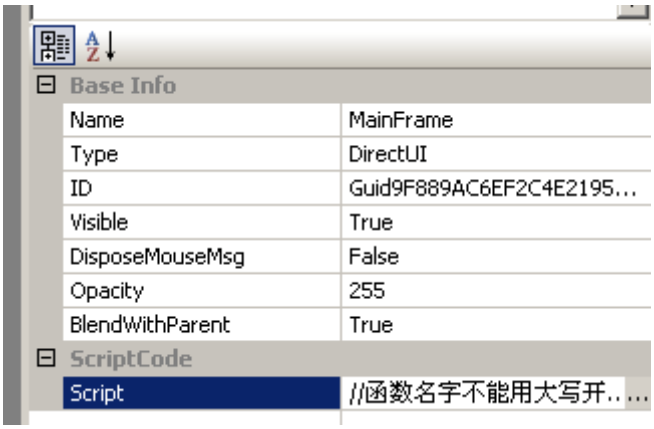
HRESULT SetItmeHBitmap([in] OLE_HANDLE hItem,[in] OLE_HANDLE hBitmap,[in] OLE_HANDLE hSelBitmap,[out,retval] VARIANT_BOOL *pbResult)
通过 hBitmap 设置 item 的图标
hItem item 的句柄 hBitmap hbitmap 句柄 hSelBitmap 选择状态的 hbitmap 句柄 pbResult 返回设置结果

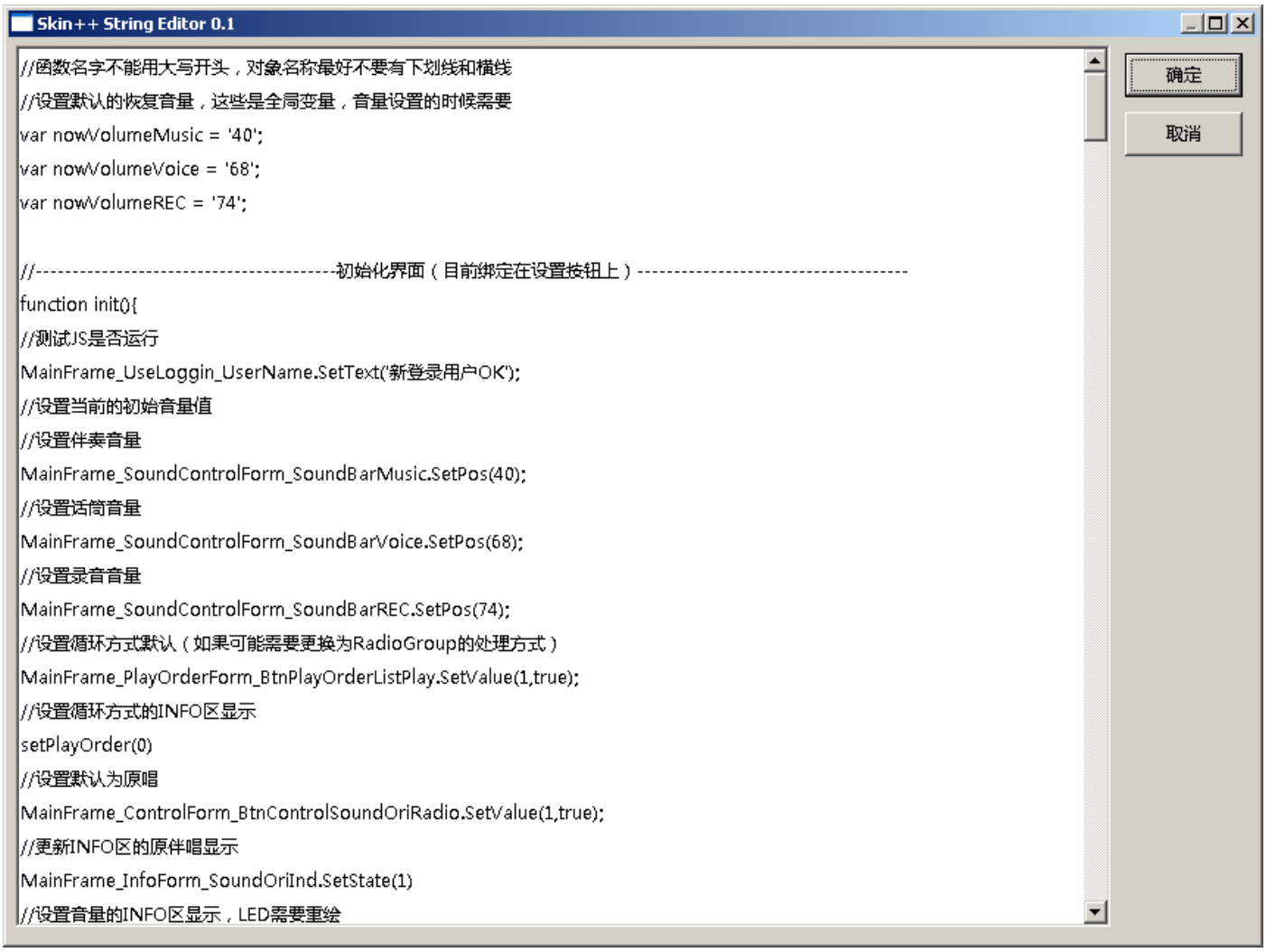
HRESULT GetItemHBitmap([in] OLE_HANDLE hItem,[out,retval] OLE_HANDLE *ppResult)
获取 item hbitmap 句柄
Hitem item 的句柄 ppResult 返回句柄

HRESULT GetItemByIndex([in]SHORT nIndex, [in]OLE_HANDLE hParent, [out,retval]OLE_HANDLE *pItem)
通过序号来得到节点
nIndex 序号 hPrant 父节点，如果为 NULL 则为根节点 pItem 返回节点句柄

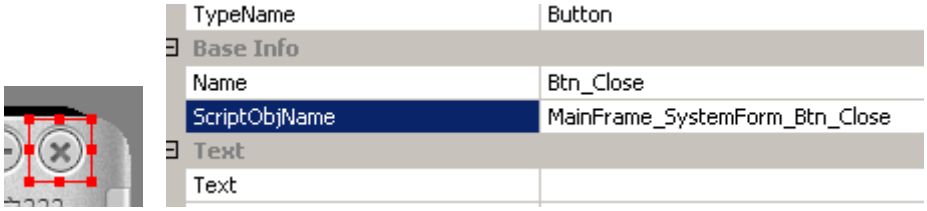
七、 利用 JavaScript 控制 DirectUI 控件对象

打开 DirectUI Builder，选择一个 DirectUI 对象，在属性区中双击 ScriptCode 即可打开当前窗口的脚本编辑窗口。





每一个皮肤控件都一个脚本对象名称，此名称是自动生成的，选择一个控件在属性区中即可看到此脚本对象名称。



通过此对象名称我们可以为这个对象添加事件。

如为一个暂定按钮添加脚本事件

```
//绑定事件到播放/暂停按钮
var strGuid = MainFrame_ControlForm_BtnControlPlayPause.GetObjectID()
Skin.AddEvent(strGuid,WM_LBUTTONUP,"playPause",true)
```

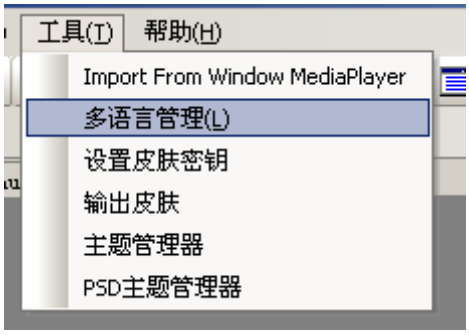
处理事件

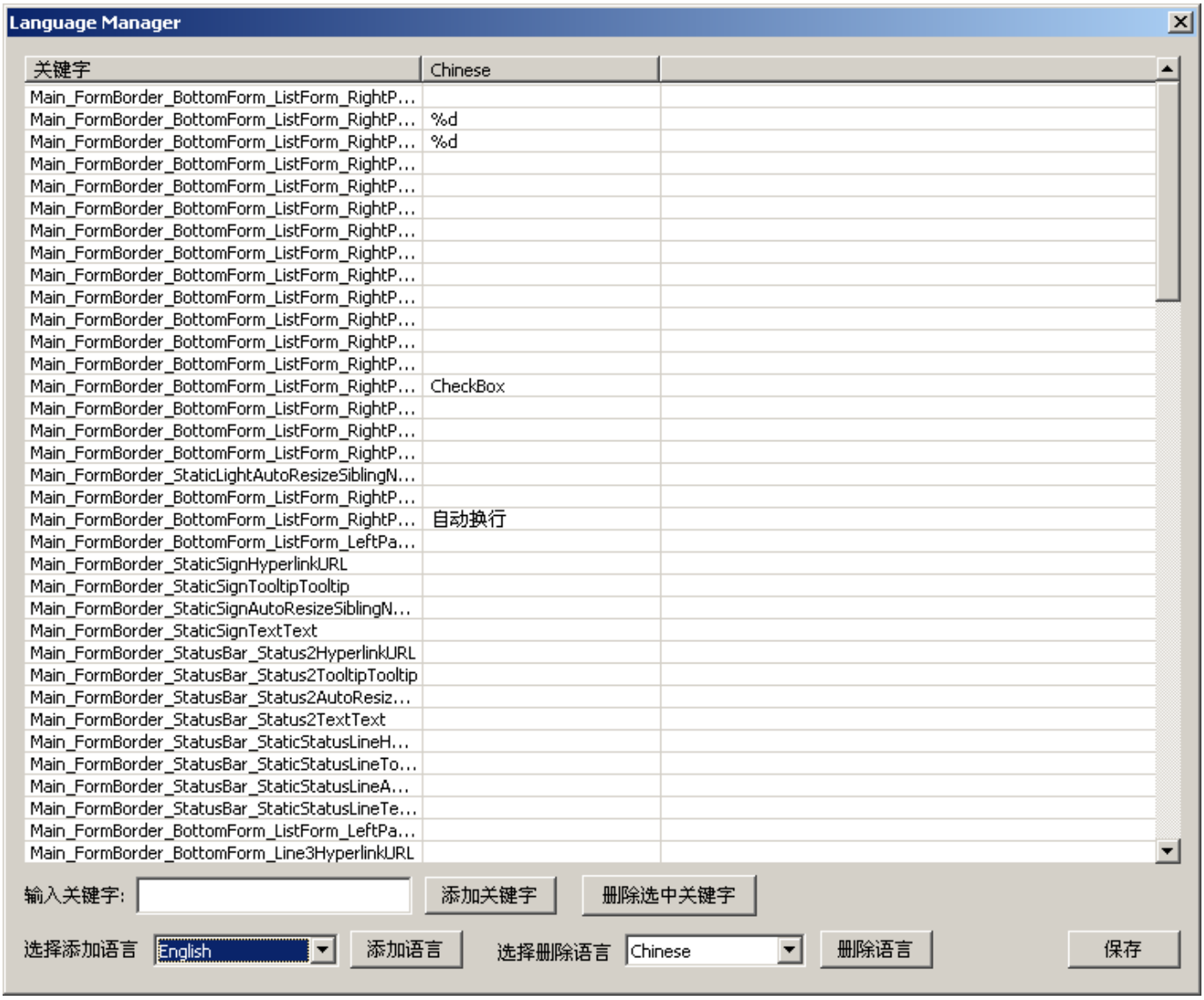
```
//----------------------------------------播放暂停（绑定主播放键盘）
    -----------------------------------
function playPause(){
//获得当前 PLAY 按钮的 VALUE，0 为 UNCHECKED 即 PAUSE 状态，1 为 CHECKED
    即 PLAY 状态
var nowState = MainFrame_ControlForm_BtnControlPlayPause.GetValue();
//获得当前录音按钮的 VALUE，判断是否在录制中播放/暂停
var recState = MainFrame_RecorderForm_BtnControlREC.GetValue();
//如果是 PLAY
if (nowState == 1){
//判断是否为录音
if(recState == 0){
//不是在录音中
//设置当前播放歌曲
MainFrame_InfoForm_SongInfo.SetText('正在播放：最熟悉的陌生人 - 萧亚轩');
//设置滚动和下一首的信息
//---------NEED TO BE DONE
//设置时间，需要 TIMER 刷新显示，要 Redraw 一下
MainFrame_InfoForm_TimeLedLength.SetValue('04:58');
MainFrame_InfoForm_TimeLedLength.DirectRedraw();
MainFrame_InfoForm_TimeLedNow.SetValue('01:24');
MainFrame_InfoForm_TimeLedNow.DirectRedraw();
}
```

所有控件的 Com 接口方法都可以利用 JavaScript 进行调用。

八、 DirectUI 多语种界面开发

界面中所有控件的界面静态文本保存在外部 XML 文件之中，可以通过 DirectUI Builder 设置不同语言的界面静态文本。使用工具，多语言管理，打开多语言设置窗口。





然后选择一个需要添加的语言，如英文 English,点击添加语言，则在列表中出现新的语言列。

键字	Chinese	English
in_FormBorderSysBtnHelpBtn		
in_FormBorderSysBtnMinBtn	MinBtn	
in_FormBorderSysBtnRestoreBtn	MaxBtn	
in_FormBorderSysBtnCloseBtn	CloseBtn	
in_FormBorderCaptionCaptionName	好压	
in_FormBorder_BottomForm_ListForm_RightP...	InfoPanel	
in_FormBorder_BottomForm_ListForm_RightP...	HwndFileList	
in_FormBorder_BottomForm_ListForm_LeftEx...	缩放左侧面板	
in_FormBorder_BottomForm_ListForm_LeftEx...		
in_FormBorder_BottomForm_ListForm_RightE...	缩放右侧面板	
in_FormBorder_BottomForm_ListForm_RightE...		

然后对文本进行逐一的翻译，最后点击“保存”按钮。

Chinese	English
MinBtn	MinBtn
MaxBtn	MaxBtn
CloseBtn	CloseBtn
好压	HaoZip

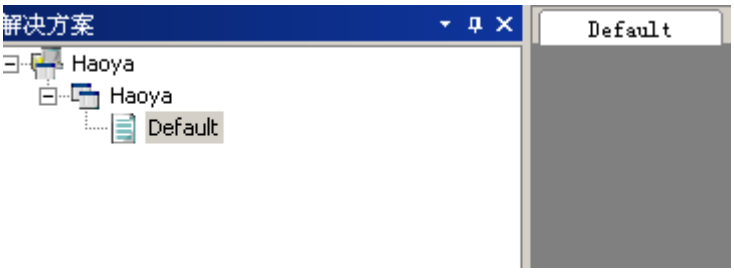
在程序运行过程中，调用 IDUIRes 的 ChangeLanguage 来在不同的语言中进行动态的切换。

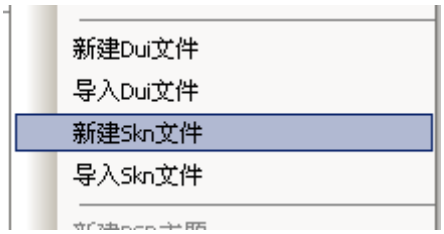
HRESULT ChangeLanguage([in] DUI_LANGUAGE eLang)
动态切换语言
eLang 显示语言的枚举值

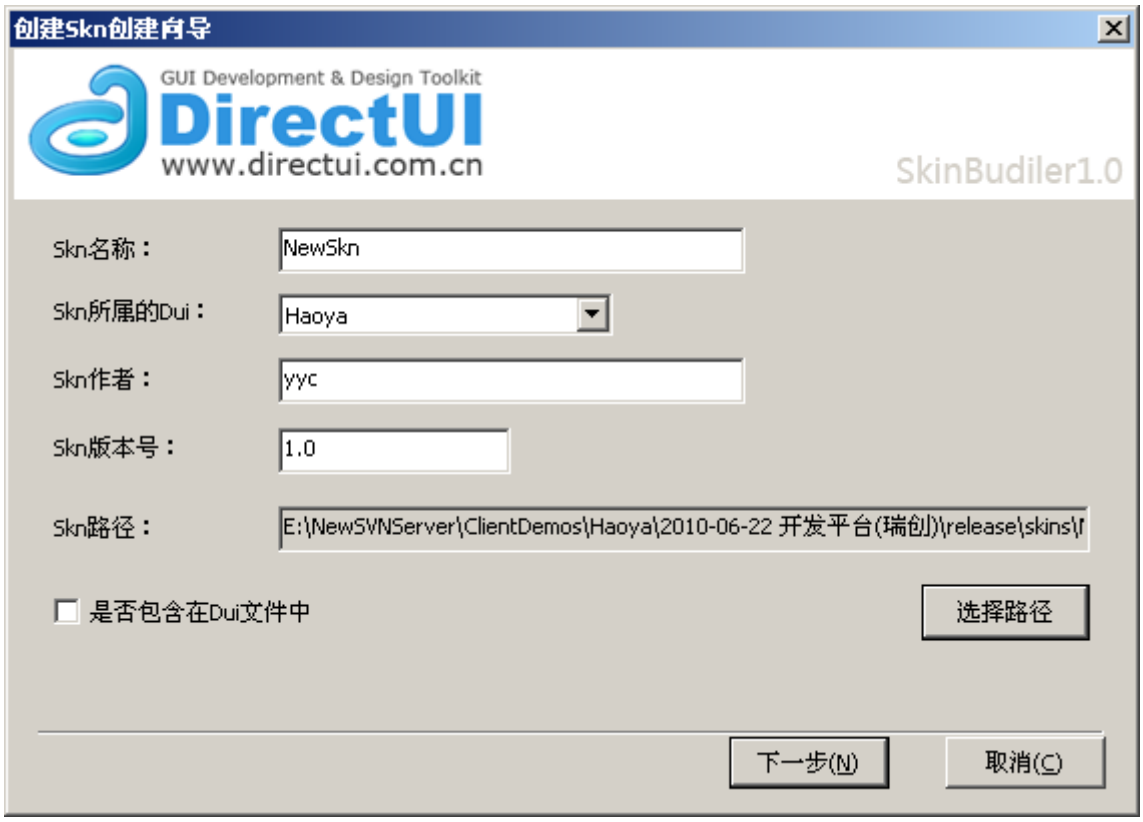
九、 DirectUI 多皮肤开发

DirectUI 支持多套皮肤的开发，每一套皮肤对应一个 skn 文件，调用 IDUIRes 的 ChangeSkn 方法，选择不同的 skn 文件即可完成程序皮肤的切换。

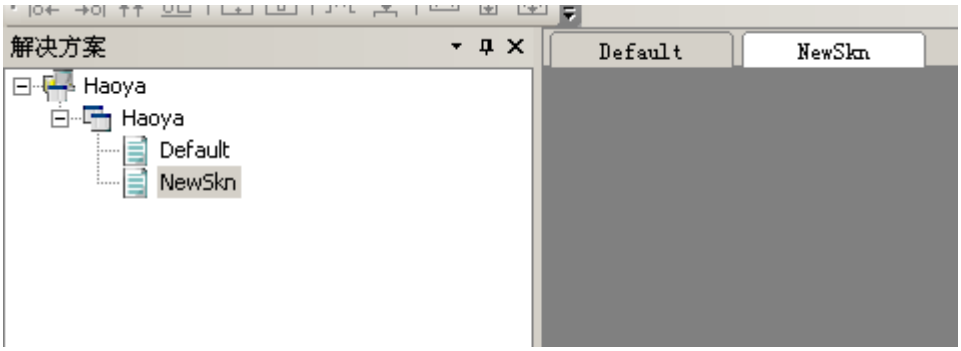
第二套 Skn 文件制作之前需要完成完成一套皮肤的制作工作，然后打开 DirectUIBuilder，使用菜单新建 skn。







完成新建后，工程取区中将出现新的 skn 节点，双击之后即可打开此 skn。



然后和之前的皮肤设置方法相同，重新设置新皮肤的外观效果。

这里有一个快捷的设置方法，可以直接拷贝之前的 skn 文件，然后通过菜单“导入 skn 文件”将拷贝的皮肤导入到解决方案之中。这样可以更加直观对皮肤进行修改。

HRESULT ChangeSkn([in]BSTR strSkinPath,[out,in] VARIANT_BOOL* pbResult
切换皮肤
strSkinPath skn 皮肤路径 pbResult 返回设置结果

十、 DirectUI 界面库扩展方法

1. 如何扩展 DirectUI 控件

由于 DirectUI 采用平台加插件的开发方式所以我们方便的对其中的控件进行扩展。开发 DirectUI 控件的步骤如下：

1） 安装控件开发向导，目前仅提供 vs2003 的开发向导。

2） 新建控件工程

3） 编写控件代码

提供的控件开发环境包含两个目录夹：

1） DUIPluginWiz2003

此目录为开发项目目录

2） DUIPluginEnvironment

为控件的编译目录夹，里面包含控件的编译依赖文件，所有的控件项目必须防止到此目录下才可以顺利编译。

2. 控件向导的安装和控件的新建

1、 打开向导文件目录 DUIPluginWiz2003

2、 打开 DUIPluginWiz2003.vsz，Param="ABSOLUTE_PATH = 当前 DUIPluginWiz2003 工程所在路径"

例如

VSWIZARD 7.0

Wizard=VsWizard.VsWizardEngine.7.1

Param="WIZARD_NAME = DUIPluginWiz2003"

Param="ABSOLUTE_PATH = E:\SVNServer\DirectUI\DirectUI\KernelControls\DUIPluginWiz2003"

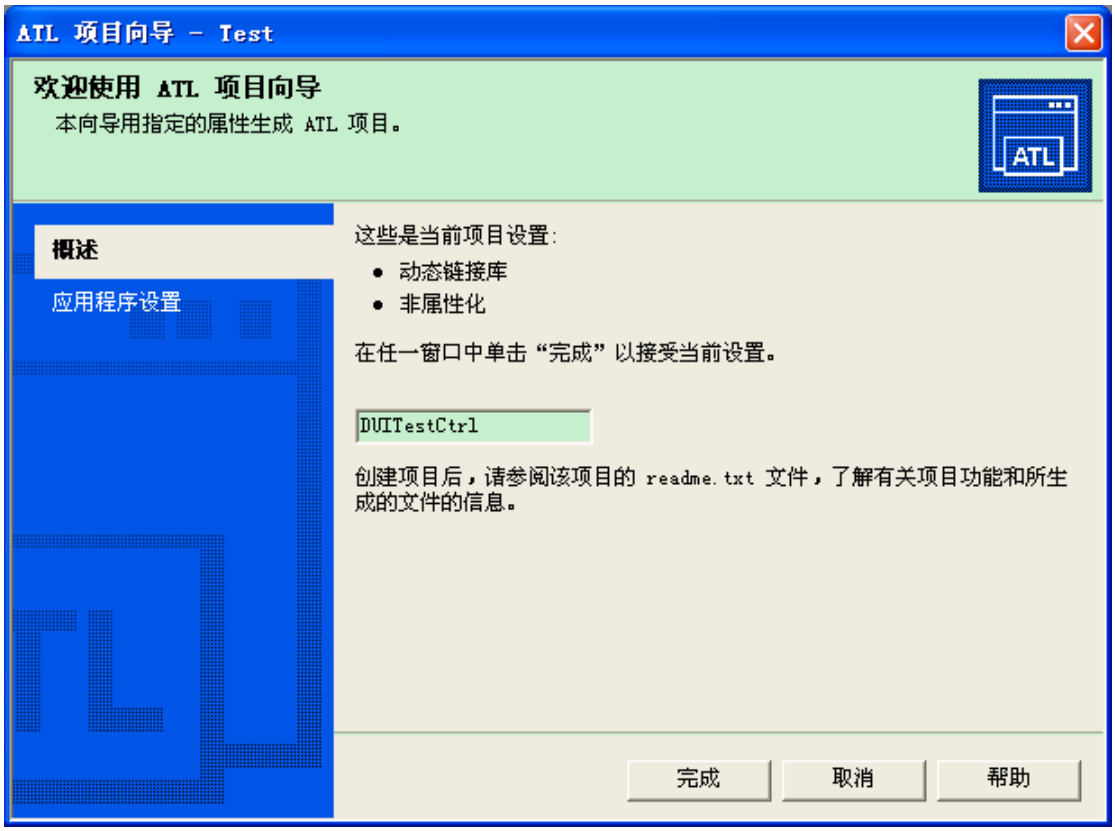
Param="FALLBACK_LCID = 2052"

3、 将 DUIPluginWiz2003.ico 和 DUIPluginWiz2003.vsz 拷贝到 Microsoft Visual Studio .NET 2003\Vc7\vcprojects 路径下

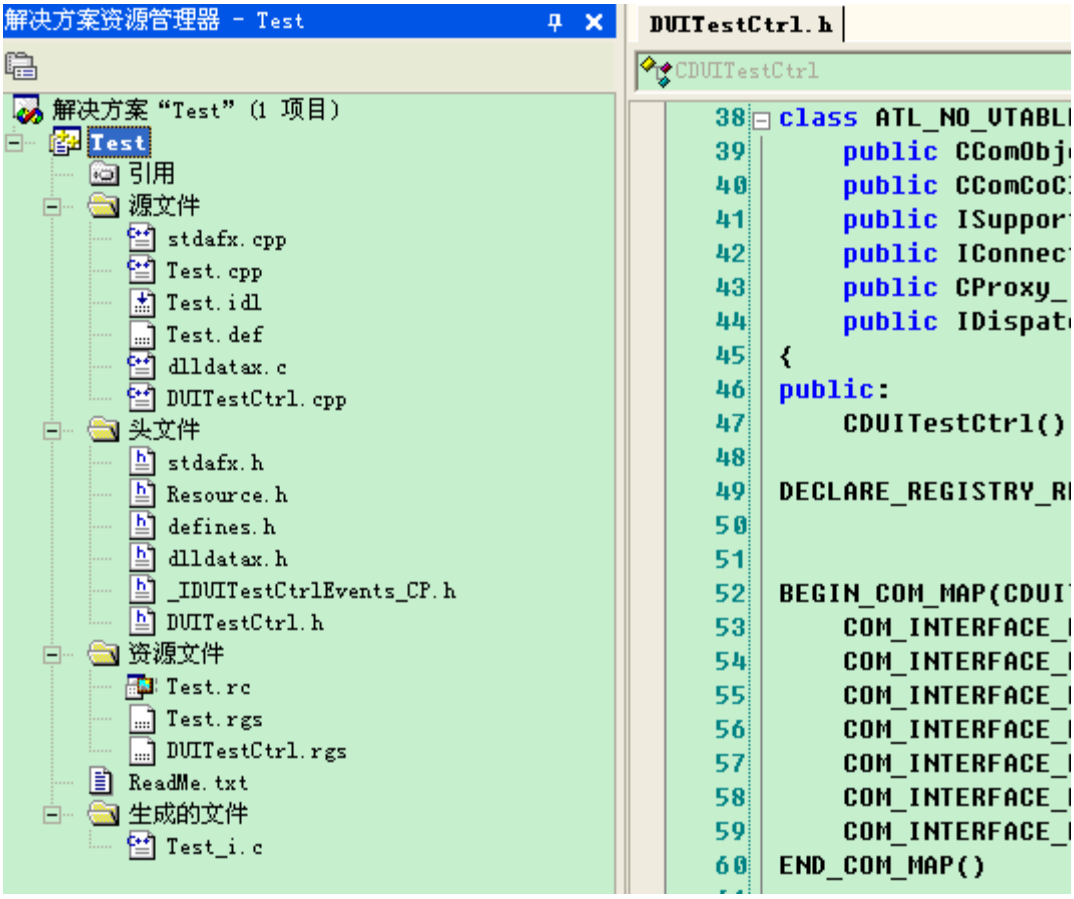
4、 打开 VS2003，新建项目中 Visual C++项目 DUIPluginWiz2003

注意保存目录必须是 DUIPluginEnvironment 目录

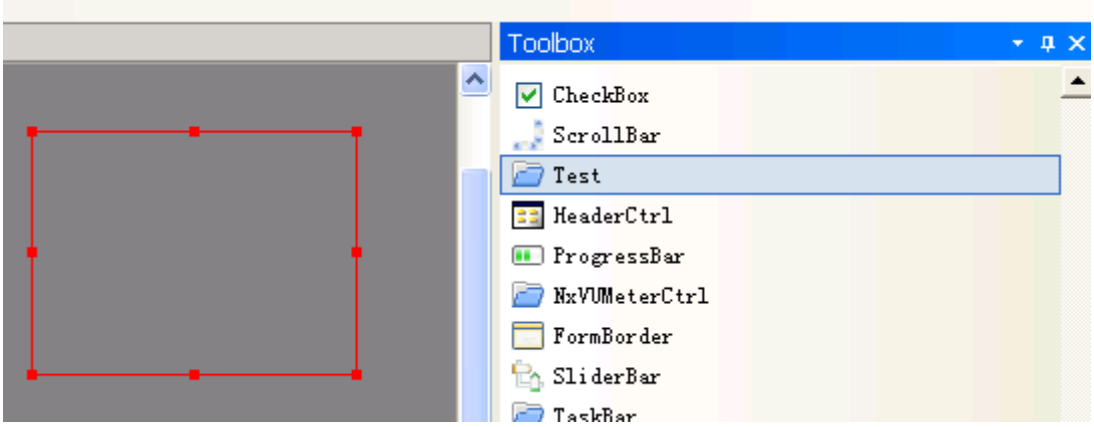
5、 分别输入工程名和控件名称，控件的名称建议使用“DUI”为其前缀。



6、 完成后我们将可以看到向导为我们建立的工程和预设代码。



7、 编译此项目后我们将会在 Builder 的工具箱中看到此控件，并可以导入到 DirectUI 窗口上。



3. 向导生成代码分析

3.1 控件文件

如新建一个新控件，工程名为 Test，控件名称为 DUITestCtrl。

向导生成如下的一个 ATL 的 COM 库，主要文件有：

1） 控件代码

DUITestCtrl.h

DUITestCtrl.cpp

2） 控件界面描述文件

Test.idl

3） 控件消息定义

Defines.h

3.2 控件代码分析

1） 控件头文件 DUITestCtrl.h

STDMETHOD(CreateProps)();

此方法为控件的属性创建方法，由平台进行调用。

STDMETHOD(DrawObject)(IDUIObjectDraw *pObjDraw,SkinRect sknrc,VARIANT_BOOL* pbResult);

此方法为控件的绘制事件。

STDMETHOD(EventNotify)(DUINotify *peVentNotify,VARIANT_BOOL *pbResult);

此方法用于接收平台向控件发送的各种消息，如 MouseMouse，MouseButton 等等。

2） 控件实现文件 DUITestCtrl.cpp

STDMETHOD(CreateProps)();

我们在此方法中创建控件的属性，这些属性可以被我们的 builder 访问到，详细的属性类型及创建请参考“控件属性类型”章节。

STDMETHOD(DrawObject)(IDUIObjectDraw *pObjDraw,SkinRect sknrc,VARIANT_BOOL* pbResult);

在该方法中我们绘制控件的外观

```
STDMETHODIMP CDUITestCtrl::DrawObject(IDUIObjectDraw *pObjDraw,
                                      SkinRect sknrc,VARIANT_BOOL* pbResult)
{
//  AFX_MANAGE_STATE(AfxGetStaticModuleState());

    UNREFERENCED_PARAMETER(sknrc);

    CSkinRect rcDraw = GetRect();

    *pbResult = VARIANT_FALSE;
    OLE_HANDLE hDCDraw = NULL;
    pObjDraw->GetDC(VARIANT_FALSE,&hDCDraw);

    *pbResult = VARIANT_TRUE;

    return S_OK;
}
```

IDUIObjectDraw* 是我们平台传入的绘制对象，类似于 CDC*对象，从 IDUIObjectDraw 中我们可以通过 GetDC 来获取 HDC。

通过控件的 GetRect 我们可以获取控件当前在此 DirectUI 窗口上的范围。

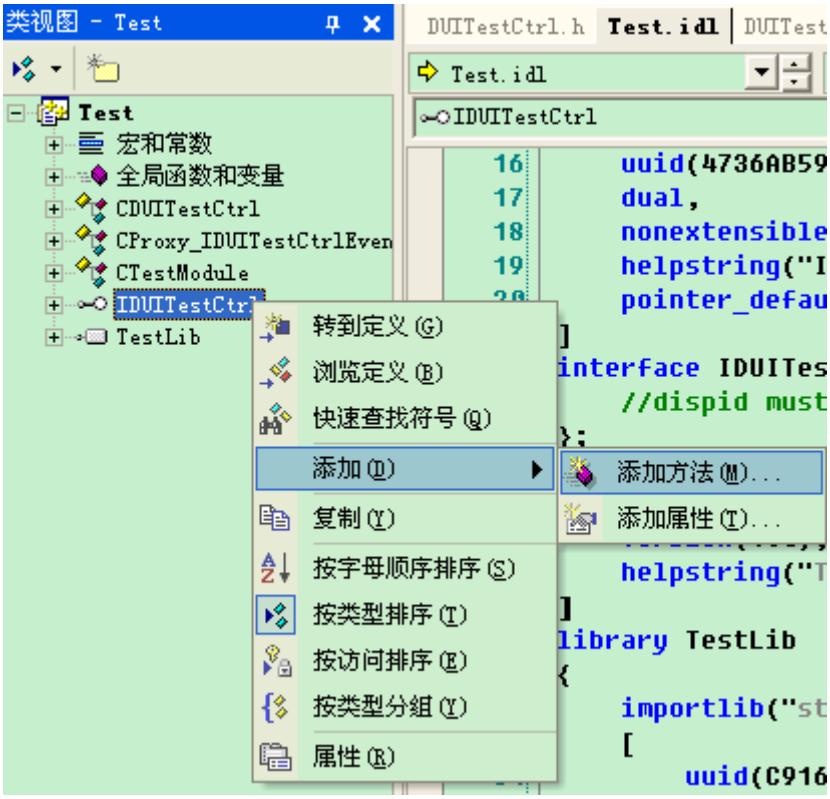
STDMETHOD(EventNotify)(DUINotify *peVentNotify,VARIANT_BOOL *pbResult);

此方法中我们接受平台抛出的消息，并对各种类型的消息进行相应。

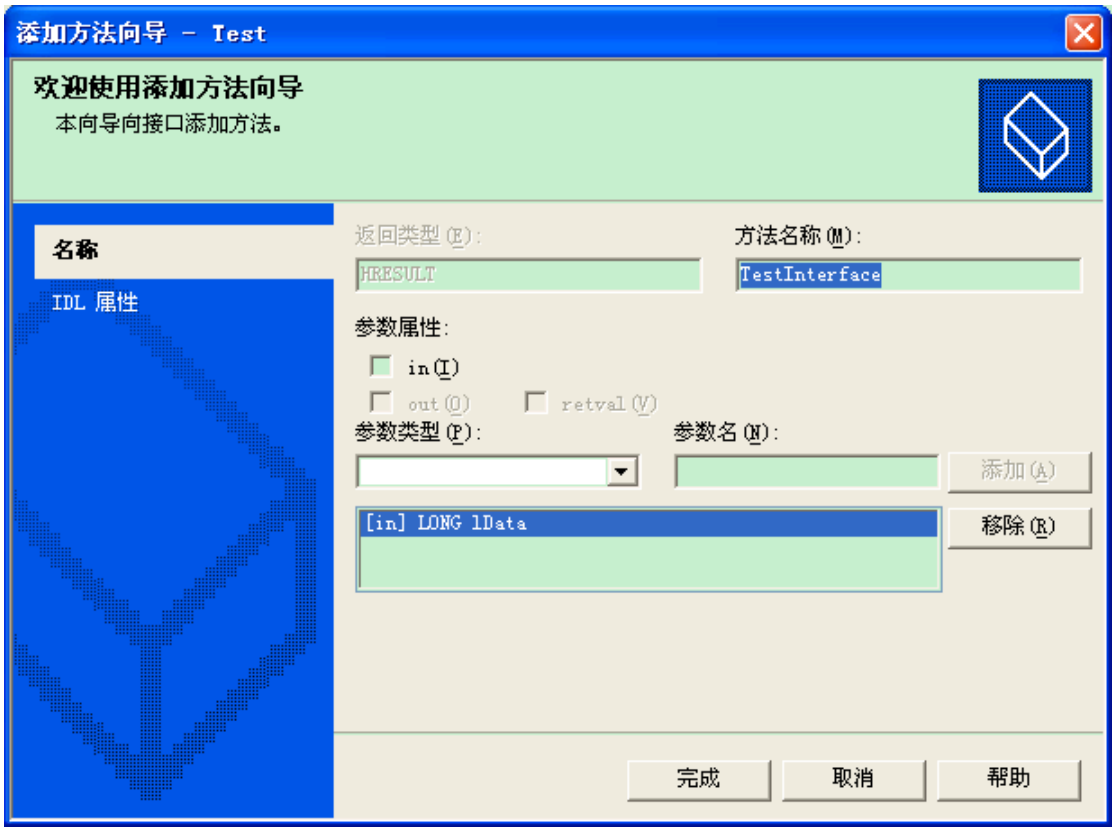
消息名称	消息功能描述
DUISM_TIMER	平台内 Timer 消息，定时器消息
DUISM_SIZE	控件大小发送改变时发出此消息，类似 WM_SIZE
DUISM_ONSETFOCUS	控件获取焦点消息
DUISM_ONKILLFOCUS	控件失去焦点消息
DUISM_MOUSELEAVE	鼠标移出控件区域的消息
DUISM_LBUTTONUP	在此控件上鼠标左键弹起消息

DUISM_LBUTTONDBCLICK	在此控件上鼠标左键双击消息
DUISM_LBUTTONDOWN	在此控件上鼠标左键按下消息
DUISM_MOUSEENTER	鼠标移入此控件消息
DUISM_MOUSEMOVE	鼠标在控件上移动消息
DUISM_DRAWPREVIEW	Builder 请求绘制预览图消息（已过时）
DUISM_PROPCHANGEDNOTIFY	控件属性在 Builder 内改变后的消息
DUISM_ISDRAGABLED	用户点击拖动此控件是否可拖动窗口的问询消息
DUISM_IMPORTCONFIG	Builder 内能否通过导入配置来创建此控件的问询消息
DUISM_FINALCREATE	控件完成创建的消息
DUISM_DESIGNSTATUSCHANGED	Builder 内控件设计状态改变消息（已过时）
DUISM_CALLPROP	属性回调消息（已过时）
DUISM_DESTROYBMPPERPIXEL	窗口销毁绘图缓存时的消息
DUISM_GETDRAGCURSOR	Builder 内问询控件拖拽的光标的消息
DUISM_GETCONTROLICON	Builder 内问询控件工具栏中显示图标的消息
DUISM_GETCATEGORYNAME:	Builder 内获取控件类型字符串的消息
UISM_GETCONTROLTYPENAME	Builder 内获取控件名称的消息
DUISM_GETAUTHORINFO	Builder 内查询控件的版本信息的消息
DUISM_INITOBJECT	控件初始化时消息
DUISM_INITDIALOG	控件所属 DirectUI 窗口初始化时消息
DUISM_ACCESSCONFIGED	控件序列化时的消息
DUISM_DESTROYING	控件所属 DirectUI 窗口销毁时的消息
DUISM_GETTABINFO	控件查询 TAB 信息的消息（已过时）
DUISM_DISPOSEMOUSEMOVE	控件是否截获平台的 MouseMouse 不向下转发的问询消息

3） 接口描述文件，为控件添加接口的方式



在项目的类视图中接口的点击右键，选择“添加函数”，弹出接口的接口添加窗口。



填写方法名称和参数列表就可以给控件暴露接口了，完成后会自定修改 Idl 和控件代码。

```
84  public:
85      DECLARE_CONTROLBASE_DEFAULT()
86
87      STDMETHOD(CreateProps)();
88      STDMETHOD(DrawObject)(IDUIObjectDraw *pO
89      STDMETHOD(EventNotify)(DUINotify *peVent
90
91      STDMETHOD(TestInterface)(LONG lData);
92  };
93
```

```
6  ////////////////////////////////////////////////////////////
7
8  STDMETHODIMP CDUITestCtrl::TestInterface(LONG lData)
9  {
0      // TODO: 在此添加实现代码
1
2      return S_OK;
3  }
4
```

4） 控件消息定义

控件对外部的消息可访问的枚举定义在 Defines.h 文件中。

```
defines.h | directuipluginsupport.h | CmdButton.h | DUITabButton.h | KernelAll.idl | DUIAnimate.h | TabCtrlI
defines.h
(全局范围)

typedef
[
    uuid(7EEF8245-6637-4890-A338-D2BD142E4346),
    version(1.0),
    helpstring("DirectUI BUTTONSTATE "),
    public
]
enum DUI_BUTTONSTATE
{
    DUI_BTN_NORMAL   = 0,//Indicates the button is normal state.
    DUI_BTN_PRESS    = 1,//Indicates the button is pressed.
    DUI_BTN_DISABLE  = 2,//Indicates the button is disable state.
    DUI_BTN_HOT      = 3,//Indicates the button is highlight state.
    DUI_BTN_FOCUS    = 4,//Indicates the button get the focus.
    DUI_BTN_LAST     = 5//Indicates the number of the button state.
}DUI_BUTTONSTATE;
```

4. 控件属性类型

属性是 Builder 中控件设置与控件的交互媒介，当我们建立一中类型的属性 Builder 中将会出现此类型的属性设置列表，当我们保存后这些属性的设置将会保存到皮肤文件中。当我们控件运行时我们的程序则可以通过这些属性获取我们之前的设置。

4.1 创建属性的方法

通过 CreateProp 方法来创建控件属性，方法参数如下：

```
CreateGroupProp(
    ICtrlPluginProp* pPropParent,      // 父控件属性节点
    enumPropType eType,                // 属性类型
    BSTR strPropName,                  // 属性名字
    BSTR strPropHelp,                  // 属性描述
    VARIANT_BOOL bIsStyle,             // 是否是样式或者普通属性，样式将出现在 Builder 的样式区中，否则将出现在 Builder 的属性区中
    IPropChangedCallback *pCallback,   // 属性设置的回调方法
    IDUIPropBase **ppProp)             // 返回属性
```

4.2 ICtrlPluginProp 属性组属性

此属性是其他属性的容器节点，用于建立属性的层次关系。

Normal	
Image	BtnLogin.Normal
Graphics	--none--
TextStyle	LoginBtn

如 Normal 节点为 ICtrlPluginProp

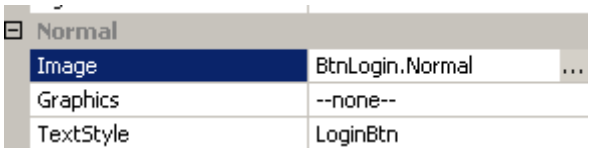
1） 声明属性

```
ICtrlPluginProp*        m_pDrawPropInactive;
```

2） CreateProps 中创建属性

```
CreateGroupProp(NULL,L"Inactive",VARIANT_TRUE,&m_pDrawPropInactive);
```

4.3 IImageSecProp ImageSection 属性



1） 声明属性

```
IImageSecProp*        m_pImgBackInactive;
```

2） CreateProps 中创建属性

```
CreateGroupProp(NULL,L"Inactive",VARIANT_TRUE,&m_pDrawPropInactive);
CreateProp(m_pDrawPropInactive,_PROPTYPE_IMAGESECTION,L"Image",L"",VARIANT_TRUE,
NULL,(IDUIPropBase**)&m_pImgBackInactive);
```

4.4 IStrProp 字符串属性

1） 声明属性

```
IStrProp*                m_pStrText;
```

2） CreateProps 中创建属性

```
CreateGroupProp(NULL,L"Text",VARIANT_TRUE,&m_pTextProp);
CreateProp(m_pTextProp,_PROPTYPE_STRING,L"Text",L"",VARIANT_FALSE,NULL,(IDUIPropBase**)&m_pStrText);
m_pStrText->SetValue(L"PushButton"); // 预设值
```

4.5 IBoolProp BOOL 属性

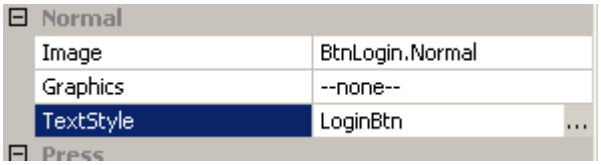
1） 声明属性

```
IBOOLProp*               m_pbShowText;
```

2） CreateProps 中创建属性

```
// Text
CreateGroupProp(NULL,L"Text",VARIANT_TRUE,&m_pTextProp);
CreateProp(m_pTextProp,_PROPTYPE_STRING,L"Text",L"",VARIANT_FALSE,NULL,(IDUIPropBase**)&m_pStrText);
m_pStrText->SetValue(L"PushButton");
CreateProp(m_pTextProp,_PROPTYPE_BOOL,L"ShowText",L"",VARIANT_FALSE,NULL,(IDUIPropBase**)&m_pbShowText);
m_pbShowText->SetValue(VARIANT_TRUE);
```

4.6 ITextStyleProp 文字样式属性



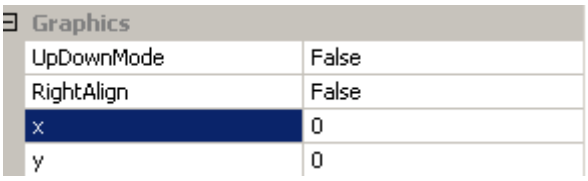
1） 声明属性

ITextStyleProp* m_pTextStyleInactive;

2） CreateProps 中创建属性

```
//Inactive
CreateGroupProp(NULL,L"Inactive",VARIANT_TRUE,&m_pDrawPropInactive);
CreateProp(m_pDrawPropInactive,_PROPTYPE_TEXTSTYLE,L"TextStyle",L"",VARIANT_TRUE,
NULL,(IDUIPropBase**)&m_pTextStyleInactive);
```

4.7 INumberLongProp 数值属性



1） 申明属性

INumberLongProp* m_pGraphX;

2） CreateProps 中创建属性

```
CreateGroupProp(NULL,L"Graphics",VARIANT_TRUE,&m_pGraphicsProp);
CreateProp(m_pGraphicsProp,_PROPTYPE_LONG,L"x",L"",VARIANT_TRUE,NULL,(IDUIPropBase**)&m_pGraphX);
```

4.8 ICursorProp 光标

1） 申明属性

ICtrlPluginProp* m_pCursorProp;

ICursorProp* m_pCursor;

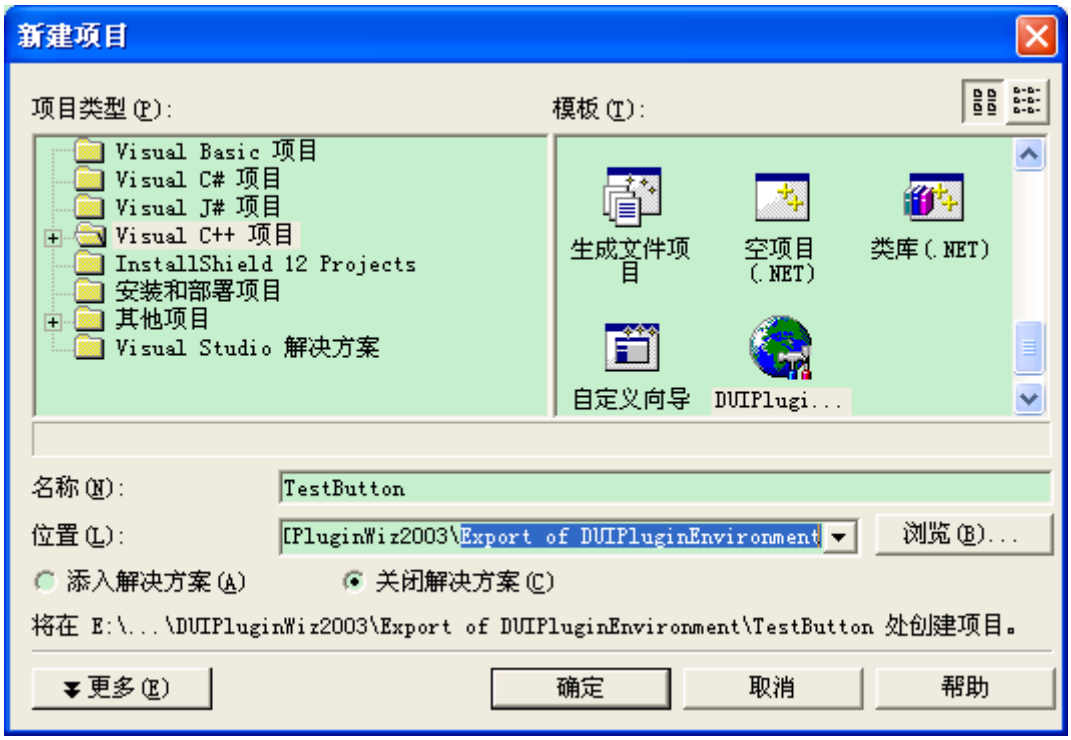
2） CreateProps 中创建属性

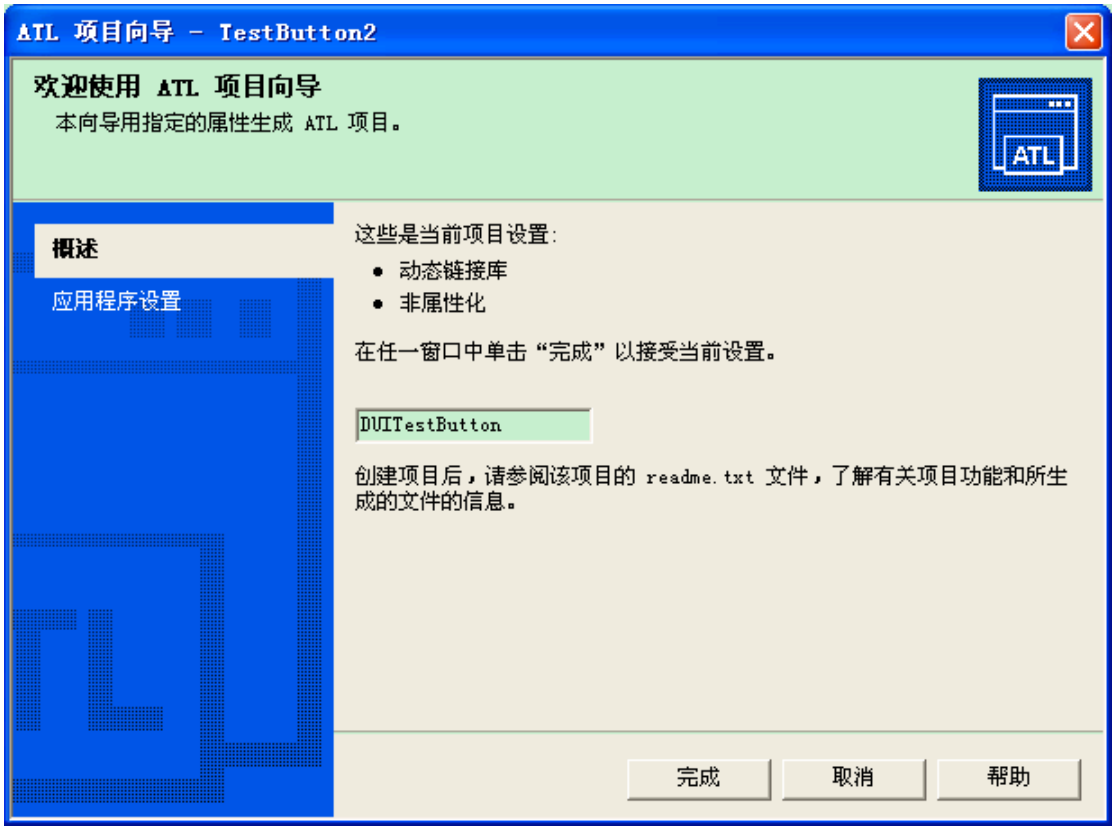
```
// Cursor
CreateGroupProp(NULL,L"Cursor",VARIANT_TRUE,&m_pCursorProp);
CreateProp(m_pCursorProp,_PROPTYPE_CURSOR,L"Cursor",L"",VARIANT_TRUE,NULL,(IDUIPropBase**)&m_pCursor);
```

5. 利用 DUI 开发向导开发一个简单的 Button 控件

5.1. 新建控件工程

1） 打开 VS2003，新建工程，选择 DUIPluginWiz 项目，工程名称是 ButtonTest，控件名称为 DUIButtonEx，保存路径为我们的控件环境目录 DUIPluginEnvironment 下。





5.2. 声明控件属性

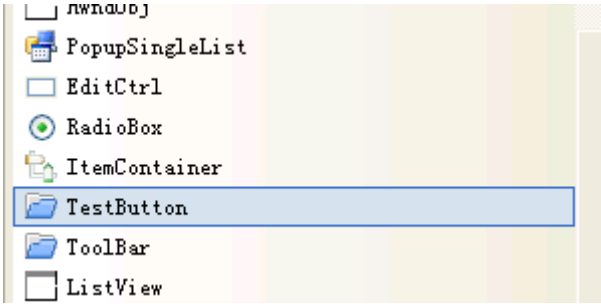
```
private:

    ////////////////////////////////////////////////////////////
    //Props

    ICtrlPluginProp*        m_pDrawProp;     // 背景属性组
    IImageSecProp*          m_pImgBack[5];   // 五个状态图片

    ICtrlPluginProp*        m_pDrawProp;     // 文本属性
    ITextStyleProp*         m_pTextStyle;    // 文字样式
    IStrProp*               m_pStrText;      // 按钮文本
```

```
ICtrlPluginProp*    m_pDrawProp;   // 背景属性组
IImageSecProp*      m_pImgBack[5];   // 五个状态图片 NormalHotPressDisableFocus
ICtrlPluginProp*    m_pDrawProp;   // 文本属性
ITextStyleProp*     m_pTextStyle;    // 文字样式
IStrProp*           m_pStrText;      // 按钮文本
```

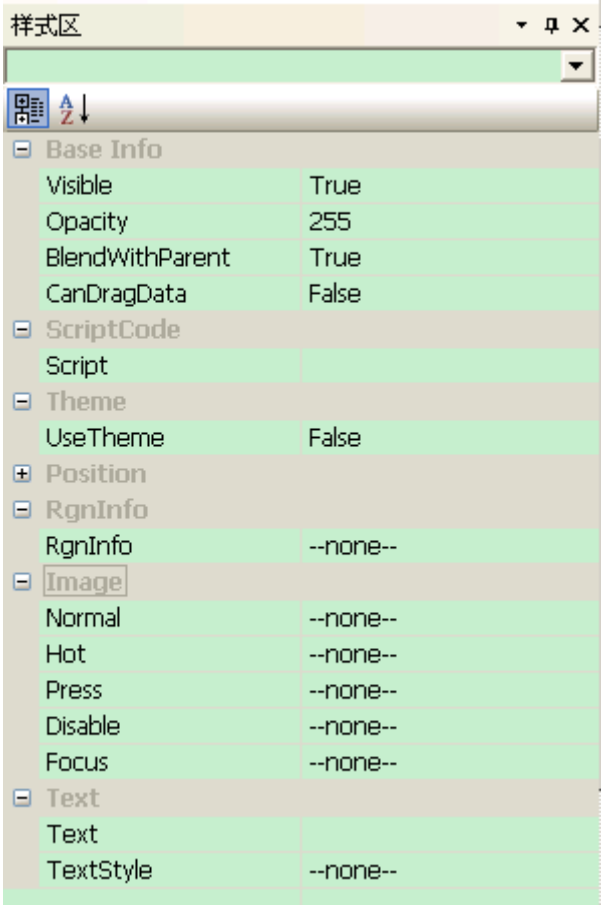
5.3. 创建控件属性

```
STDMETHODIMP CDUITestButton::CreateProps()
{
    // 创建图片属性组
    CreateGroupProp(NULL, L"Image", VARIANT_FALSE, &m_pDrawProp);
    CreateProp(m_pDrawProp, _PROPTYPE_IMAGESECTION, L"Normal", L"Normal", VARIANT_TRUE, NULL, (IDUIPropBase ** )&m_pImgBack[0]);
    CreateProp(m_pDrawProp, _PROPTYPE_IMAGESECTION, L"Hot", L"Hot", VARIANT_TRUE, NULL, (IDUIPropBase ** )&m_pImgBack[1]);
    CreateProp(m_pDrawProp, _PROPTYPE_IMAGESECTION, L"Press", L"Press", VARIANT_TRUE, NULL, (IDUIPropBase ** )&m_pImgBack[2]);
    CreateProp(m_pDrawProp, _PROPTYPE_IMAGESECTION, L"Disable", L"Disable", VARIANT_TRUE, NULL, (IDUIPropBase ** )&m_pImgBack[3]);
    CreateProp(m_pDrawProp, _PROPTYPE_IMAGESECTION, L"Focus", L"Focus", VARIANT_TRUE, NULL, (IDUIPropBase ** )&m_pImgBack[4]);

    // 创建文字属性组
    CreateGroupProp(NULL, L"Text", VARIANT_FALSE, &m_pTextProp);
    CreateProp(m_pTextProp, _PROPTYPE_STRING, L"Text", L"Text", VARIANT_TRUE, NULL, (IDUIPropBase ** )&m_pStrText);
    CreateProp(m_pTextProp, _PROPTYPE_TEXTSTYLE, L"TextStyle", L"TextStyle", VARIANT_TRUE, NULL, (IDUIPropBase ** )&m_pTextStyle);

    return S_OK;
}
```

编译运行后我们将在 Builder 的工具箱中看到 TestButton 控件

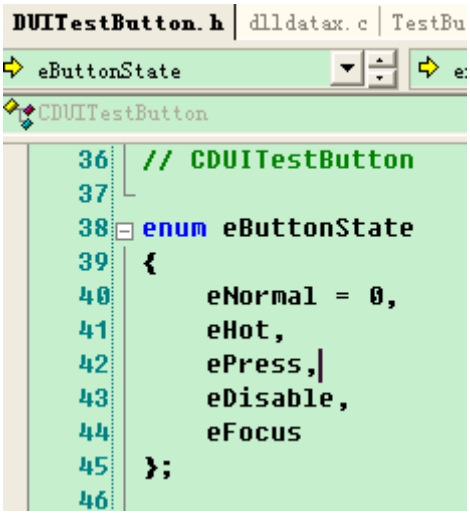


建立一个 DirectUI 窗口后拖入此控件，我们在样式区中看到此控件的属性。



5.4. 处理控件绘图

1） 控件状态枚举



枚举当前控件处于何种状态下，增加成员变量

eButtonState m_eButtonState;

2） 修改 DrawObject 方法

```
STDMETHODIMP CDUITestButton::DrawObject(IDUIObjectDraw *pObjDraw,
                                        SkinRect sknrc,VARIANT_BOOL* pbResult)
{
    UNREFERENCED_PARAMETER(sknrc);

    CSkinRect rcDraw = GetRect();

    *pbResult = VARIANT_FALSE;
    OLE_HANDLE hDCDraw = NULL;
    pObjDraw->GetDC(VARIANT_FALSE,&hDCDraw);

    // hDCDraw 获取的 DC 的原点处于此控件的左上角
    // 根据状态绘制背景
    rcDraw.OffsetRect(-rcDraw.TopLeft().x, -rcDraw.TopLeft().y);
    VARIANT_BOOL bResult = VARIANT_FALSE;
    m_pImgBack[m_eButtonState]->Draw(hDCDraw, rcDraw, &bResult);

    // 获取文本属性中的数据
    BSTR strText;
    m_pStrText->GetValue(&strText);
    m_pTextStyle->Draw(hDCDraw, rcDraw, strText, &bResult);

    *pbResult = VARIANT_TRUE;

    return S_OK;
}
```

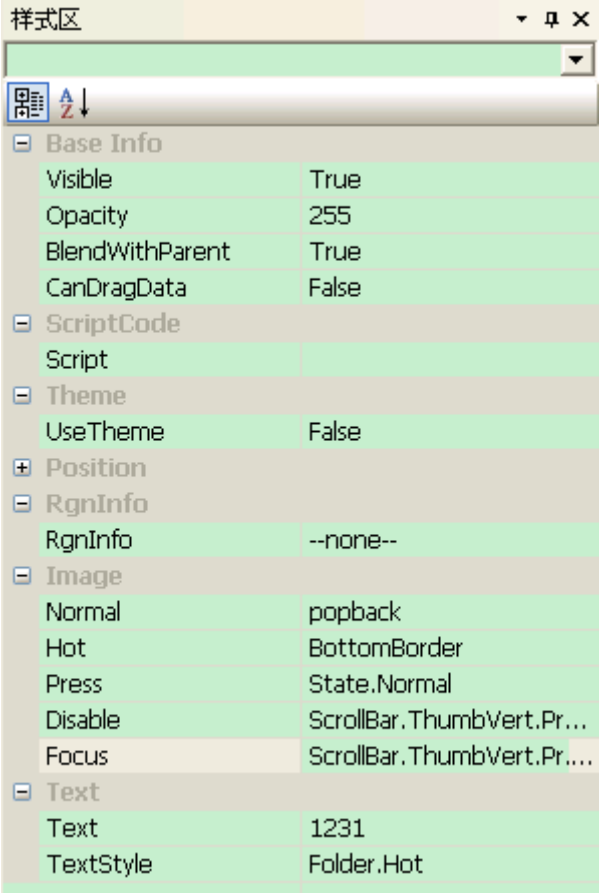
5.5. 处理控件消息响应

```
    case DUISM_MOUSELEAVE:
```

```
            {
                LONG* pnResult = (LONG*)peVentNotify->lParam1;

                If（m_eButtonState！ = eNormal）
                {
                m_eButtonState = eNormal;
                DirectRedraw();
}

                *pbResult = VARIANT_FALSE;
            }
            break;

        case DUISM_LBUTTONDOWN:
            {
                LONG nHitTest = peVentNotify->lParam1;
                LONG x = peVentNotify->lParam2;
                LONG y = peVentNotify->lParam3;
                LONG* pnResult = (LONG*)peVentNotify->lParam4;

                if（m_eButtonState != ePress）
                {
                    m_eButtonState = ePress;
                    DirectRedraw();
                }

                *pbResult = VARIANT_FALSE;
            }
            break;

        case DUISM_MOUSEENTER:
            {
                LONG nHitTest = peVentNotify->lParam1;
                LONG x = peVentNotify->lParam2;
                LONG y = peVentNotify->lParam3;
                LONG* pnResult = (LONG*)peVentNotify->lParam4;

                if（m_eButtonState != eHot）
                {
                    m_eButtonState = eHot;
                    DirectRedraw();
                }

                *pbResult = VARIANT_FALSE;
            }
            break;
```

5.6. Builder 设置

编译完成后我们打开 Builder，对此控件进行属性设置，之后测试运行就可以看到此控件的运行效果了。



十一、 DirectUI 使用 FAQ

1. 概要问题

1.1 如何联系我们

DirectUI 为上海勇进软件公司旗下产品

DirectUI 产品网站：http://www.directui.com

上海勇进软件网站：http://www.uipower.com

联系电话：

86-021-34021068（总机）

86-021-34021068-817（技术）
86-021-34021068-808（销售）
电子邮箱：Sales@uiower.com

2. 设计问题

2.1 为什么需要对切图进行处理

由于 ImageBase 支持图片的九宫拉伸所以可以去除一些不必要的像素，所以需要对切图进行处理，除去不必要的像素。

2.2 Budiler 更新后无法启动，提示异常

大部分情况下出现的原因是之前的控件库没有反注册掉

3. Builder 使用问题

3.1 启动 Builder 提示“TIMEOUT 查询函数”失败，如何处理

出现该问题的原因是硬件狗授权时间到期，请联系上海勇进软件公司延长授权时间。

4. 界面库开发问题

3.1 VC++6.0 下编译提示“提示 USES_CONVERSION 未定义”

请包含<atlconv.h>

3.2 如何在多线程项目调用界面库接口

DirectUI 库的控件接口在多线程中调用可能出现同步异常，所以需要在多线程通过通过自定义消息将请求发送到界面上，界面根据响应此消息，此响应方法中调用界面库接口即可。