Windows Phone Mango编程实践

***Windows Phone Mango Programming Practice***

第一篇 基础篇

# 应用程序栏(Application Bar)最佳实践

日月逝矣，岁不我与。本章主要讲解应用程序栏(Application Bar)的设计及其本地化的实践，介绍应用Visual Studio的无缝结合的工具——Expression Blend炫彩Silverlight页面。

## 应用程序栏（Application Bar）介绍

应用程序栏（Application Bar）提供软件开发者放置程序最常使用的指令，最多可以同时放置四个图标按钮。如图5-1 Application Bar。

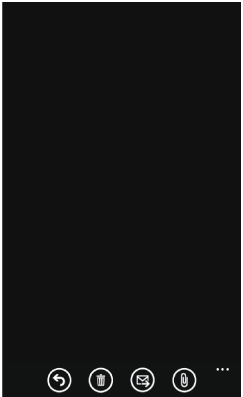


图5-1 Application Bar

应用程序栏默认显示没有文字叙述的按钮，且点击状态栏右侧「…」或利用手指直接从应用程序栏向上滑动可以显示完整的应用程序栏。

Windows Phone为应用程序栏(Application Bar)提供了菜单动画和旋转支持。不论横向视图还是纵向视图，应用程序栏都会在屏幕固定的位置上显示—自动延伸到靠近屏幕按键的那端显示。按钮图像也会随着屏幕方向而转向（除了不支持180度纵向，其余三个方向皆支持）。

应用程序栏的按钮可以设定成启用或无法使用状态。比如我们制作一个电子阅读器的软件，在电子书是只读的情况下，可以设定删除功能的按钮无效，即无法使用删除按钮。

使用原则：

1. 应用程序栏的应遵循愈精简愈好的原则。
2. 建议使用系统默认的主题配色即可。

使用自定的颜色可能会影响到按钮图案的质量以及会产生功耗方面的负影响。

1. 应用程序栏的透明度可以调整，建议其数值为0、0.5、1。

请注意，如果设定值小于1，应用程序栏将会覆盖屏幕上的主画面，如果设定值为1，主画面的显示大小则会产生变化。

1. 应用程序栏的图标按钮的图像必须清晰明确、容易理解，请以用户熟悉的图例来表达。
2. 每个图标按钮都必须有图案和文字叙述，文字叙述愈精简愈好。
3. Windows 手机都有专用的硬件后退按钮，因此不要在应用程序栏中设置后退导航按钮。

如果应用程序栏还不能满足您的设计需要，您还想在应用程序栏展现更多的快捷方式，那使用应用程序栏(Application Bar)的扩展功能—应用程序栏菜单(Application Bar Menu)。

应用程序栏菜单（Application Bar Menu）是从应用程序栏执行特定指令的选择性方式。启动应用程序栏菜单的方式可由按下右侧的「…」按键，或利用手指直接从应用程序栏向上滑动。关闭的方式是点击菜单以外的区域、使用退回键，或是按下菜单中的按键，则可离开应用程式列菜单。应用程序栏菜单启动后将一直停留在屏幕视图上，直到使用者有给予指令关闭为止。

为了防止菜单的滚动，菜单上最多只能显示五项指令，如图5-2 应用程序栏菜单(Application Bar Menu)。从用户体验的角度，设计应用程序栏应遵循的第一原则就是：愈精简愈好，充分体现Metro的风格。

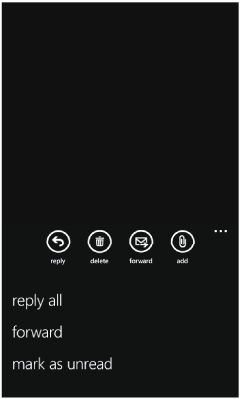


图5-2 应用程序栏菜单(Application Bar Menu)

应用程序栏不是 Silverlight 控件，并且不支持数据绑定。这意味着用于按钮标签的字符串值必须在 XAML 中的硬编码，并且不能本地化。如果您计划将应用程序本地化，应通过C# 语言的编码实现创建应用程序栏，并使用C#语言以编程方式实现本地化。

对于应用程序栏的设计的动手实践中，我们还会用到一个强大的设计工具Microsoft Expression Blend 4。 Expression Blend和Visual Studio的无缝结合，将为我们呈现亦幻亦真的Windows Phone界面开发。



图5-3 Expression Blend

## 动手实践——设计应用程序栏

我们使用两种方式创建应用程序栏，第一种采用C#编程语言创建，第二种[http://i.msdn.microsoft.com/Hash/030c41d9079671d09a62d8e2c1db6973.gif](javascript:void(0))在XAML中创建。

### [[http://i.msdn.microsoft.com/Hash/030c41d9079671d09a62d8e2c1db6973.gif](javascript:void(0))添加图标按钮图标的图像](javascript:void(0))

应用程序栏中的图标按钮的图像在使用前，必须将它们添加到您在 Visual Studio 中的项目中。添加方法：

1. 为图像文件创建一个子目录，在[Solution Explorer]解决方案资源管理器中，右键单击您的项目名称并选择[**Add**]，然后选择[**New Folder**]。重命名文件夹为"images"。
2. 在 Windows 资源管理器中，将图标图像复制到图像文件夹images中。
3. 在 Visual Studio的[**Solution Explorer**]解决方案资源管理器中右键单击图片文件夹images，选择[**Add**]，然后[**Existing Item…**.]。选择一个您的图像或按住 ctrl 键选择多个图像，然后单击[**Add**]。
4. 对于每个图像，在[**Solution Explorer**]解决方案资源管理器中的右键单击图像，然后选择**[Properties]**。
5. 将[Build Action]属性设置为" Content "，将[**Copy to Output]**属性设置 “Copy always”。

### [C#创建应用程序栏](javascript:void(0))

如果您在C#中编写代码比使用XAML的更舒适，亦或者应用程序要实现本地化功能，那么您可以将应用程序栏的添加通过C#语言编程来实现，而无需编辑XAML文件。

在本例ApplicationBarSample工程中，打开MainPage.xaml.cs文件，在文件头部添加using指令。

Silverlight Project: ApplicationBarSample File: MainPage.xaml.cs

using Microsoft.Phone.Shell;

1. 在构造函数中，初始化ApplicationBar、

public MainPage()

{

InitializeComponent();

//支持的横向视图和纵向视图

this.SupportedOrientations = SupportedPageOrientation.PortraitOrLandscape;

1. 设定ApplicationBar的显示属性IsVisible为true（显示），应用程序栏菜单(Application Bar Menu)的显示属性IsMenuEnabled为true（有效）。

ApplicationBar = new ApplicationBar();

ApplicationBar.IsMenuEnabled = true;

ApplicationBar.IsVisible = true;

ApplicationBar.Opacity = 1.0;

1. 创建ApplicationBarIconButton 对象，并分配该按钮的 Click 事件的事件处理程序。

请注意，必须为图标按钮的[Text](http://www.microsofttranslator.com/bv.aspx?from=&to=zh-CHS&a=http%3A%2F%2Fmsdn.microsoft.com%2Fzh-cn%2Flibrary%2Fmicrosoft.phone.shell.applicationbariconbutton.text.aspx)属性分配一个字符串。否则程序在运行时会引发 InvalidOperationException的异常。

ApplicationBarIconButton hide = new ApplicationBarIconButton(new Uri("/Images/expand.png", UriKind.Relative));

hide.Text = "hide";

hide.Click += new EventHandler(hide\_Click);

ApplicationBarIconButton opacity = new ApplicationBarIconButton(new Uri("/Images/opacity.png", UriKind.Relative));

opacity.Text = "opacity";

opacity.Click += new EventHandler(opacity\_Click);

ApplicationBarIconButton enabled = new ApplicationBarIconButton(new Uri("/Images/menuenabled.png", UriKind.Relative));

enabled.Text = "enabled";

enabled.Click += new EventHandler(enabled\_Click);

1. 调用ApplicationBar.Buttons.Add方法，将图标按钮添加到应用程序栏的集合中。

ApplicationBar.Buttons.Add(hide);

ApplicationBar.Buttons.Add(opacity);

ApplicationBar.Buttons.Add(enabled);

1. 构造应用程序栏菜单(Application Bar Menu)

创建ApplicationBarMenuItem对象，构造函数的参数为显示菜单项名称的字符串。然后指定的菜单项的Click事件的事件处理程序。

ApplicationBarMenuItem foregroundItem = new ApplicationBarMenuItem("use foreground color");

foregroundItem.Click += new EventHandler(foregroundItem\_Click);

ApplicationBarMenuItem accentItem = new ApplicationBarMenuItem("use accent color");

accentItem.Click += new EventHandler(accentItem\_Click);

1. 在应用程序栏的 MenuItems 集合中添加菜单项。

ApplicationBar.MenuItems.Add(foregroundItem);

ApplicationBar.MenuItems.Add(accentItem);

UpdateText();

}

1. 在图标按钮和菜单项的 Click事件处理程序中编写代码。

Silverlight Project: ApplicationBarSample File: MainPage.xaml.cs

/// <summary>

/// Click handler for accent color menu item.

/// Changes the colored UI elements to the built-in PhoneAccentColor

/// </summary>

/// <param name="sender">The control that raised the click event.</param>

/// <param name="e">An EventArgs object containing event data.</param>

void accentItem\_Click(object sender, EventArgs e)

{

UpdateColor((Color)Resources["PhoneAccentColor"]);

}

/// <summary>

/// Click handler for accent color menu item.

/// Changes the colored UI elements to the built-in PhoneForegroundColor

/// </summary>

/// <param name="sender">The control that raised the click event.</param>

/// <param name="e">An EventArgs object containing event data.</param>

void foregroundItem\_Click(object sender, EventArgs e)

{

UpdateColor((Color)Resources["PhoneForegroundColor"]);

}

/// <summary>

/// Click handler for opacity icon button.

/// Sets the opacity value of the ApplicationBar to 0, 1, or .5

/// </summary>

/// <param name="sender">The control that raised the click event.</param>

/// <param name="e">An EventArgs object containing event data.</param>

void opacity\_Click(object sender, EventArgs e)

{

if (ApplicationBar.Opacity < .01)

{

ApplicationBar.Opacity = 1;

}

else if (ApplicationBar.Opacity > .49 && ApplicationBar.Opacity < .51)

{

ApplicationBar.Opacity = 0;

}

else

{

ApplicationBar.Opacity = .5;

}

UpdateText();

}

/// <summary>

/// Click handler for hide icon button.

/// Changes the Visible property of the ApplicationBar to false

/// And makes the "Show Application Bar" button visible

/// </summary>

/// <param name="sender">The control that raised the click event.</param>

/// <param name="e">An EventArgs object containing event data.</param>

void hide\_Click(object sender, EventArgs e)

{

ApplicationBar.IsVisible = false;

showButton.Visibility = Visibility.Visible;

UpdateText();

}

/// <summary>

/// Click handler for menu enable icon button.

/// Changes the IsMenuEnabled property of the ApplicationBar

/// When IsMenuEnabled is false, the menu will not pop up

/// </summary>

/// <param name="sender">The control that raised the click event.</param>

/// <param name="e">An EventArgs object containing event data.</param>

void enabled\_Click(object sender, EventArgs e)

{

ApplicationBar.IsMenuEnabled = !ApplicationBar.IsMenuEnabled;

UpdateText();

}

/// <summary>

/// Click handler for show button.

/// Sets the Visible property of the Application Bar to true

/// </summary>

/// <param name="sender">The control that raised the click event.</param>

/// <param name="e">An EventArgs object containing event data.</param>

private void showButton\_Click(object sender, RoutedEventArgs e)

{

ApplicationBar.IsVisible = true;

showButton.Visibility = Visibility.Collapsed;

UpdateText();

}

/// <summary>

/// Updates the TextBlock objects to reflect the current state

/// of the ApplicationBar

/// </summary>

void UpdateText()

{

VisibleTextBlock.Text = ApplicationBar.IsVisible ? "yes" : "no";

OpacityTextBlock.Text = ApplicationBar.Opacity.ToString("0.0");

MenuEnabledTextBlock.Text = ApplicationBar.IsMenuEnabled ? "yes" : "no";

}

/// <summary>

/// Helper method for changing the color of the UI

/// </summary>

/// <param name="c">The new color for the UI elements</param>

void UpdateColor(Color c)

{

SolidColorBrush brush = new SolidColorBrush(c);

VisibleTextBlock.Foreground = brush;

OpacityTextBlock.Foreground = brush;

MenuEnabledTextBlock.Foreground = brush;

((LinearGradientBrush)Resources["Gradient"]).GradientStops[1].Color = c;

}

### [[http://i.msdn.microsoft.com/Hash/030c41d9079671d09a62d8e2c1db6973.gif](javascript:void(0))在XAML中创建应用程序栏](javascript:void(0))

在应用程序中，完全可以在XAML中将应用程序栏添加到页面中。但是，应用程序栏不是 Silverlight 控件，并且不支持数据绑定。这意味着用于按钮标签的字符串值必须在 XAML 中的硬编码，并且不能本地化。

在XAML中创建应用程序栏的过程非常简单，Windows Phone的应用程序栏模板已经在MainPage.xaml文件中了，您需要做的全部就是取消应用程序栏定义的注释，并修改代码以使用您的图标按钮的图像和菜单选项的文字。若要添加事件处理程序的按钮和菜单项，将光标置于该元素的名称后直接并键入一个空格，然后开始键入该事件的名称，Visual Studio的智能感知功能将协助您完成对于事件处理程序的添加，您所关注的重点只要是处理逻辑部分即可，其他的事情交给智能感知去完成吧。

下面是应用程序栏的默认代码。

Silverlight Project: ApplicationBarSample File: MainPage.xaml

<!--Sample code showing usage of ApplicationBar-->

<phone:PhoneApplicationPage.ApplicationBar>

<shell:ApplicationBar IsVisible="True" IsMenuEnabled="True">

<shell:ApplicationBarIconButton Click="opacity\_Click" IconUri="/Images/appbar\_button1.png" Text="Button 1"/>

<shell:ApplicationBarIconButton IconUri="/Images/appbar\_button2.png" Text="Button 2"/>

<shell:ApplicationBar.MenuItems>

<shell:ApplicationBarMenuItem Text="MenuItem 1"/>

<shell:ApplicationBarMenuItem Text="MenuItem 2"/>

</shell:ApplicationBar.MenuItems>

</shell:ApplicationBar>

</phone:PhoneApplicationPage.ApplicationBar>

## 动手实践——本地化应用程序栏([Localizing an Application Bar](javascript:void(0)))

为ApplicationBarSample工程创建本地化的应用程序栏，即为应用程序栏添加多语言的支持。

### 添加资源文件以实现对本地化的支持

使用Visual Studio打开ApplicationBarSample工程，按以下步骤添加资源文件。

1. 在**[Solution Explorer]**解决方案资源管理器中，右键单击项目名称ApplicationBarSample并选择[**Add**]，然后选择[**New Item**]，如图5-4 Add New Item。

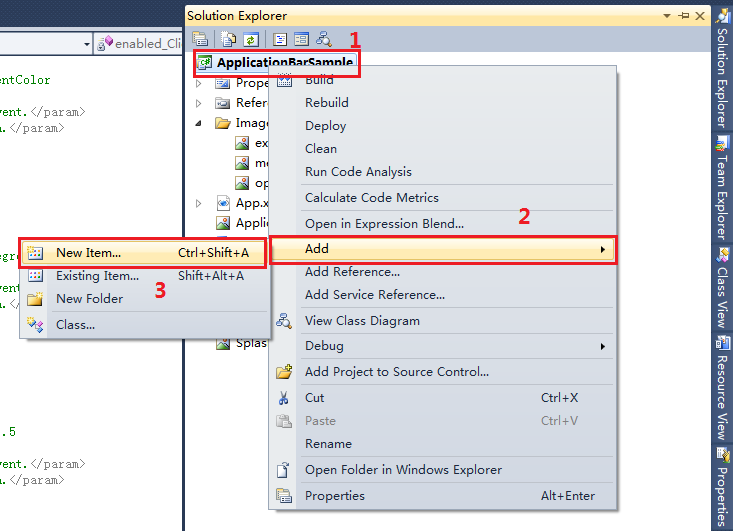


图5-4 Add New Item

1. 在 [**Add New Item** ]对话框中选择**[Resources File]**，修改文件名称为AppResources.resx。此文件将包含应用程序的默认语言的资源，如图5-5 添加资源文件AppResources。

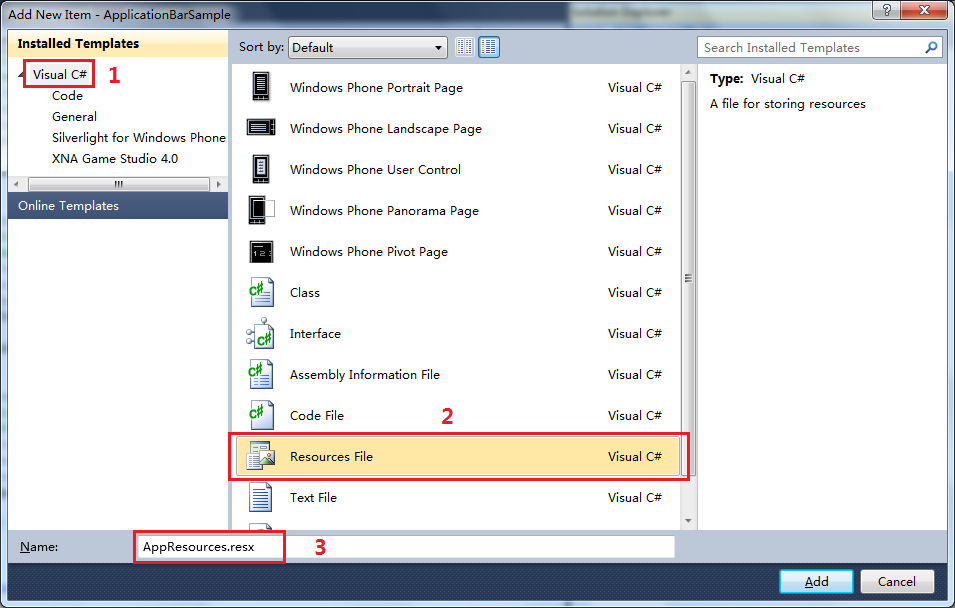


图5-5 添加资源文件AppResources

1. 确定您的应用程序中的字符串，并将它们添加到资源文件。您可以输入一个名称、 值和每个字符串的可选注释。
   1. 该名称必须是唯一的。请尽可能详细描述。
   2. 将在应用程序中向用户显示的字符串值。
   3. 注释是可选的。在大的资源文件或者字符串很多的情况下，注释会发挥很大的作用。
2. 添加其他每种语言的资源文件。

本例中支持西班牙语的资源文件AppResources.es-ES.resx和支持德语的资源文件AppResources.de-DE.resx，请直接导入到工程中即可。您也可以制作其他语言的资源文件，本例中我们的重点在于实现多语言显示的功能上。

### 定义默认的区域

1. 在**[Solution Explorer]**解决方案资源管理器中，右键单击项目名称ApplicationBarSample并选择**[Properties]**属性。
2. 在Application选项卡中，点击**[Assembly Information]**按钮。
3. 在[**Neutral Language**]列表中，选择默认的区域性。此标识语言的默认资源文件中的字符串。例如，如果默认资源文件被命名为**AppResources.resx**，并在该文件中的字符串支持英语（美国）语言，则可以选择**English (United States)** ，作为项目的中立国语言（**Neutral Language** ）。

预定义的区域性名称的完整列表如下。

|  |  |
| --- | --- |
| Culture | Culture/language name |
| English (United States) | en-US |
| English (Canada) | en-CA |
| English (Great Britain) | en-GB |
| French (France) | fr-FR |
| French (Canada) | fr-CA |
| French (Belgium) | fr-BE |

表5-1 区域语言列表

### 其他区域性语言

关闭Visual Studio，然后使用文本编辑器打开的项目文件 (**ApplicationBarSample.csproj**)。

找到 <SupportedCultures> 标记，并添加其他区域性 （语言），以满足应用程序需要。

语言名称使用分号分隔，如下所示：

Silverlight Project: ApplicationBarSample File: ApplicationBarSample.csproj

<SupportedCultures>

de-DE;es-ES

</SupportedCultures>

### [资源文件的字符串](javascript:void(0))替换

[本节实现硬编码的字符串替换资源文件中的字符串](javascript:void(0))，使用Visual Studio重新打开ApplicationBarSample工程。

1. 在**[Solution Explorer]**解决方案资源管理器中打开资源文件AppResources.resx，在**[AccessModifier]**列表框中选择**[Public]**，如图5-6 **AccessModifier**列表框。对每个项目中的资源文件，请重复此步骤。

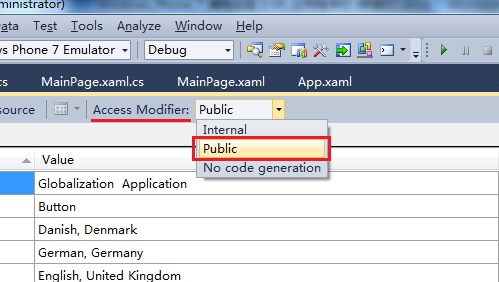


图5-6 **AccessModifier**列表框

1. 添加具有指向资源的属性的类LocalizedStrings.。在下面的示例中， LocalizedStrings 类包含一个指向 AppResources GlobalizationSample 命名空间中的资源文件的属性。

Silverlight Project: ApplicationBarSample File: LocalizedStrings.cs

namespace ApplicationBarSample

{

public class LocalizedStrings

{

public LocalizedStrings()

{

}

private static ApplicationBarSample.AppResources localizedresources = new ApplicationBarSample.AppResources();

public ApplicationBarSample.AppResources Localizedresources { get { return localizedresources; } }

}

}

1. 打开App.xaml文件添加如下的代码到<Application.Resources>标签。

Silverlight Project: ApplicationBarSample File: App.xaml

<!--Application Resources-->

<Application.Resources>

<local:LocalizedStrings xmlns:local ="clr-namespace:ApplicationBarSample"

x:Key="LocalizedStrings" />

</Application.Resources>

### 本地化应用程序栏

下面的代码示例显示一个生成应用程序栏，使用提供的字符串值的本地化的资源。

1. 创建应用程序栏

创建Application Bar的部分，显示AppResources的文字信息。

Silverlight Project: ApplicationBarSample File: MainPage.xaml.cs

#region Initialization

/// <summary>

/// Constructor for the PhoneApplicationPage

/// The ApplicationBar is initialized. Icon buttons and menu items are added

/// to the ApplicationBar and event handlers are set.

/// </summary>

public MainPage()

{

InitializeComponent();

this.SupportedOrientations = SupportedPageOrientation.PortraitOrLandscape;

ApplicationBar = new ApplicationBar();

ApplicationBar.IsMenuEnabled = true;

ApplicationBar.IsVisible = true;

ApplicationBar.Opacity = 1.0;

ApplicationBarIconButton hide = new ApplicationBarIconButton(new Uri("/Images/expand.png", UriKind.Relative));

//hide.Text = "hide";

//显示AppResources的文字信息

hide.Text = AppResources.ButtonText;

hide.Click += new EventHandler(hide\_Click);

ApplicationBarIconButton opacity = new ApplicationBarIconButton(new Uri("/Images/opacity.png", UriKind.Relative));

//opacity.Text = "opacity";

//显示AppResources的文字信息

opacity.Text = AppResources.ButtonText;

opacity.Click += new EventHandler(opacity\_Click);

ApplicationBarIconButton enabled = new ApplicationBarIconButton(new Uri("/Images/menuenabled.png", UriKind.Relative));

//enabled.Text = "enabled";

enabled.Text = AppResources.ButtonText;

enabled.Click += new EventHandler(enabled\_Click);

ApplicationBar.Buttons.Add(hide);

ApplicationBar.Buttons.Add(opacity);

ApplicationBar.Buttons.Add(enabled);

//ApplicationBarMenuItem foregroundItem = new ApplicationBarMenuItem("use foreground color");

ApplicationBarMenuItem foregroundItem = new ApplicationBarMenuItem(AppResources.MenuItemText);

foregroundItem.Click += new EventHandler(foregroundItem\_Click);

//ApplicationBarMenuItem accentItem = new ApplicationBarMenuItem("use accent color");

ApplicationBarMenuItem accentItem = new ApplicationBarMenuItem(AppResources.MenuItemText);

accentItem.Click += new EventHandler(accentItem\_Click);

ApplicationBar.MenuItems.Add(foregroundItem);

ApplicationBar.MenuItems.Add(accentItem);

UpdateText();

}

#endregion

1. 选择语言的列表

在XAML中增加选择语言的列表.

Silverlight Project: ApplicationBarSample File: MainPage.xaml

<TextBlock Height="30" Name="textBlock1" Text="{Binding Path=Localizedresources.TextLabelLocale, Source={StaticResource LocalizedStrings}}" Width="443" Foreground="{StaticResource PhoneAccentBrush}" Margin="6,7,6,573" />

<ListBox Height="94" HorizontalAlignment="Left" Margin="21,43,0,0" Name="locList" VerticalAlignment="Top" Width="441" SelectedIndex="-1" SelectionChanged="LocList\_SelectedIndexChanged" Grid.Row="1">

<ListBoxItem Content="{Binding Path=Localizedresources.LangRegionNameDe, Source={StaticResource LocalizedStrings}}" FontSize="22" />

<ListBoxItem Content="{Binding Path=Localizedresources.LangRegionNameEs, Source={StaticResource LocalizedStrings}}" FontSize="22" />

<ListBoxItem Content="{Binding Path=Localizedresources.LangRegionNameEnUS, Source={StaticResource LocalizedStrings}}" FontSize="22"/>

</ListBox>

添加后的预览效果，如图5-7 MainPage，出现选择三种语言的列表：德语、西班牙语和英语。

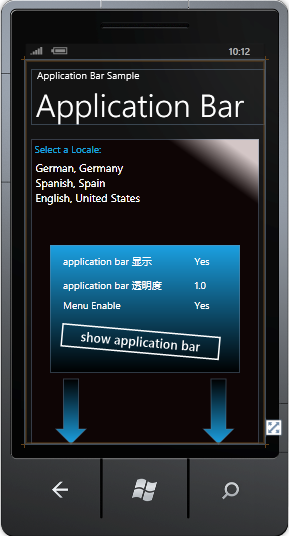


图5-7 MainPage

1. 列表的事件响应

当手指触控列表选择的语言种类发生改变时，事件响应程序LocList\_SelectedIndexChanged开始运行，修改页面上显示的文字，以及应用程序栏菜单(Application Bar Menu)中显示的文字。

至此，本地化的工作完成。

Silverlight Project: ApplicationBarSample File: MainPage.xaml.cs

private void LocList\_SelectedIndexChanged(object sender, SelectionChangedEventArgs e)

{

// Set the current culture according to the selected locale and display information such as

// date, time, currency, etc in the appropriate format.

string nl;

string cul;

nl = locList.SelectedIndex.ToString();

switch (nl)

{

case "0":

cul = "es-ES";

break;

case "1":

cul = "de-DE";

break;

case "2":

cul = "en-US";

break;

default:

cul = "en-US";

break;

}

// set this thread's current culture to the culture associated with the selected locale

CultureInfo newCulture = new CultureInfo(cul);

Thread.CurrentThread.CurrentCulture = newCulture;

CultureInfo cc, cuic;

cc = Thread.CurrentThread.CurrentCulture;

cuic = Thread.CurrentThread.CurrentUICulture;

VisibleLabel.Text = cc.NativeName;

VisibleTextBlock.Text = "";

//OpacityLabel.Text = cuic.DisplayName;

OpacityLabel.Text = "";

OpacityTextBlock.Text = "";

MenuEnabledLabel.Text = "";

MenuEnabledTextBlock.Text = "";

//localize icon button text

if (this.ApplicationBar.Buttons != null)

{

foreach (ApplicationBarIconButton btn in this.ApplicationBar.Buttons)

{

btn.Text = cc.NativeName.Substring(0, cc.NativeName.ToString().Length/2);

}

}

//localize menu buttons text

if (this.ApplicationBar.MenuItems != null)

{

foreach (ApplicationBarMenuItem itm in this.ApplicationBar.MenuItems)

{

itm.Text = cc.NativeName;

}

}

}

### 运行结果

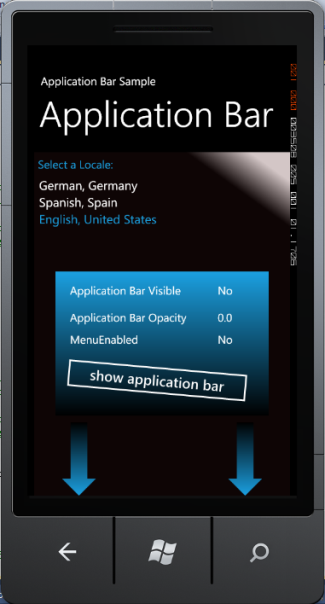


图5-8 启动界面

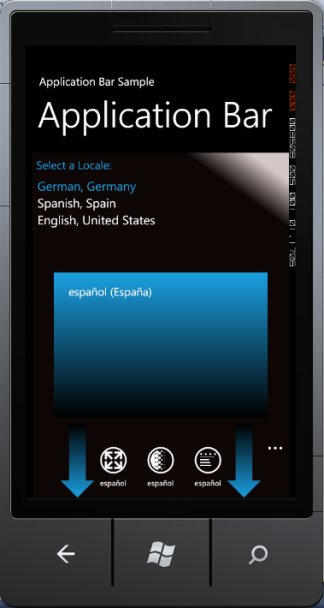
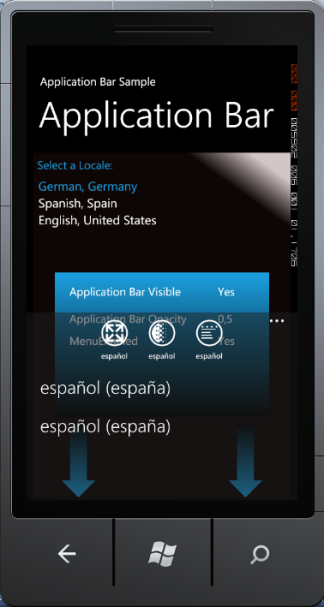
 

图5-9 德语界面和德语本地化的应用程序栏

## 应用Expression Blend炫彩Silverlight页面

请看下面的两个画面的对比图。

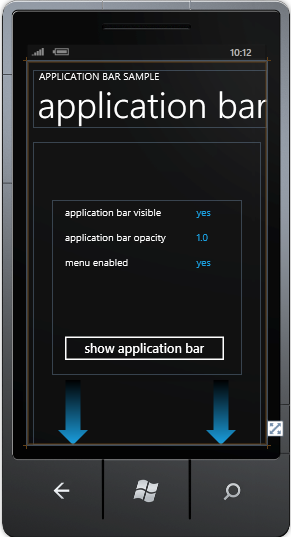
 

图5-10 对比图

在右侧的画面中，我们对左侧的画面做了三处修改，使得普通画面变得好看且很炫。如果我的手机里有这样一个很酷很炫的应用程序，我会Show给朋友们看看。

对比两个画面，修改的三处分别是背景颜色变为渐变；中间面板的颜色变为蓝色渐变，与箭头的风格保持一致；按钮被旋转了。这些变化如果通过手工修改代码那就太复杂了，使用设计工具Microsoft Expression Blend 4，轻松取代复杂的代码编写。

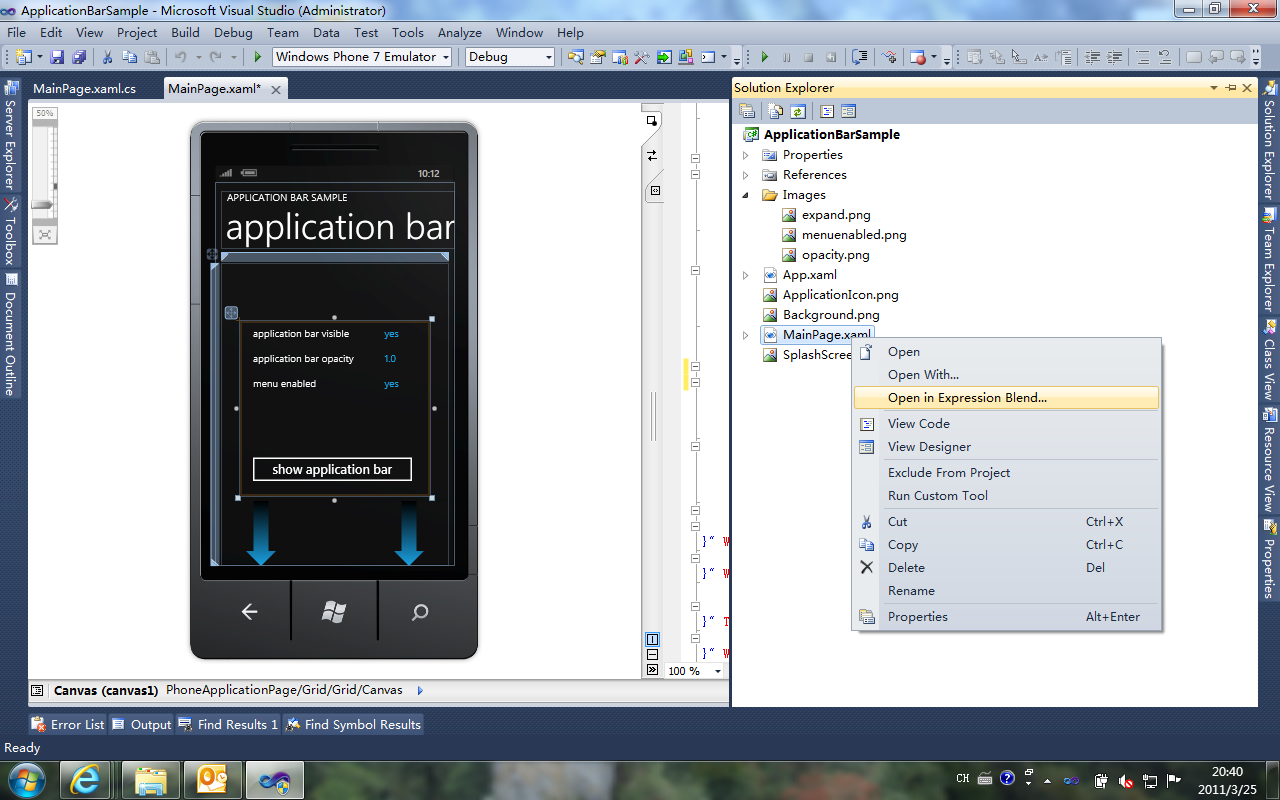
开发人员的重心永远在代码的逻辑，设计方面的创意交给设计人员思考。在软件制作工程中，如果身兼开发和设计两项职责，那在设计时就应转变思路。Windows Phone的页面设计工作，交给Microsoft Expression Blend 4完成吧，我是程序员，我需要时间陪家人喝茶聊天。

日月逝矣，岁不我与。毫不迟疑，即刻动手，使用Expression Blend动手修改呆板的画面吧。

### Expression Blend的应用

1. 启动Expression Blend 4

在Visual Studio 2010的**[Solution Explorer]**中右键点击MainPage.xaml文件，选择**[Open in Expression Blend…]**。如图5-11 启动Expression Blend 4。



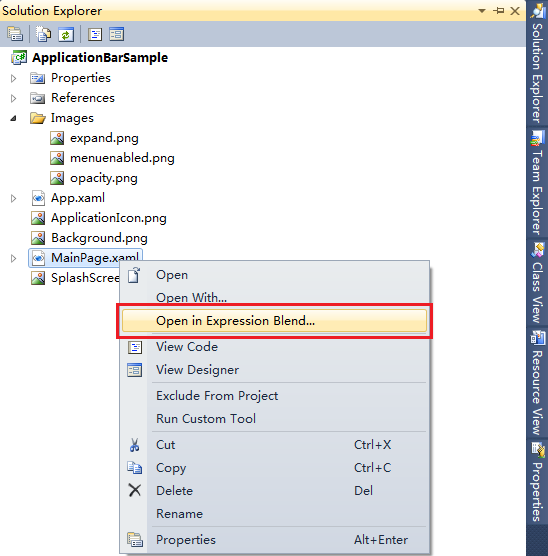


图5-11 启动Expression Blend 4

出现下图时，请点击**[Yes]**，进入Expression Blend启动画面。

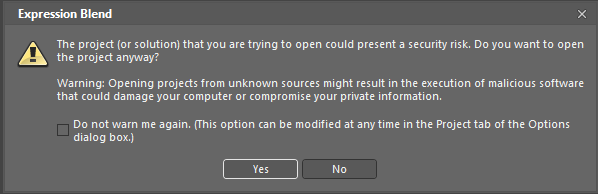


图5-12 Expression Blend

1. 画面修改一

选择**[Gradient Tool]**，设置背景颜色变为渐变，如图5-13 Gradient Tool和图5-14 设定渐变。

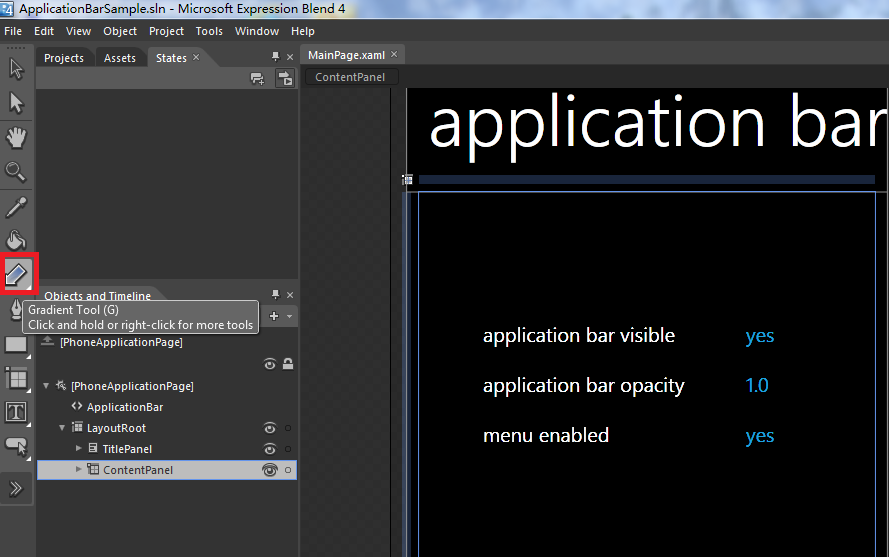


图5-13 Gradient Tool



图5-14 设定渐变

1. 画面修改二

中间面板的颜色变为蓝色渐变，与箭头的风格保持一致，如图5-15 设定中间面板的颜色变为蓝色渐变。

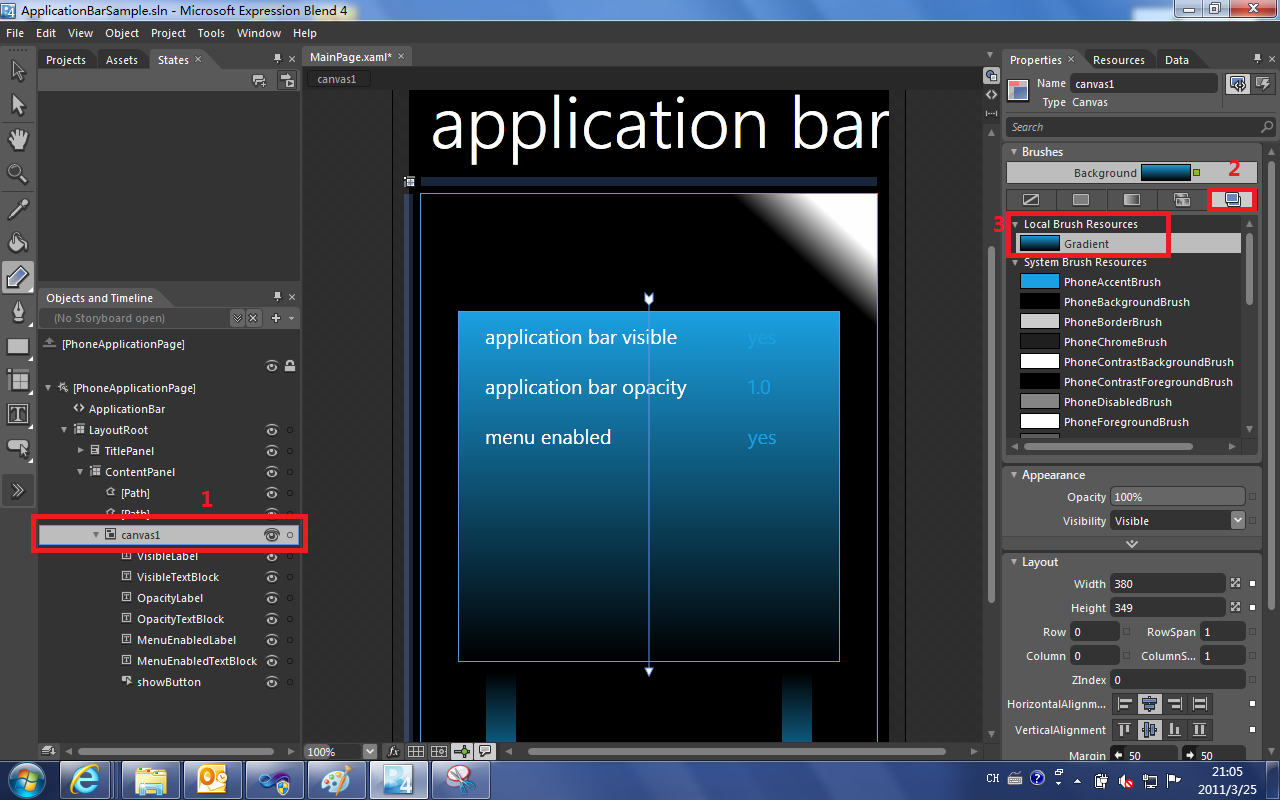


图5-15 设定中间面板的颜色变为蓝色渐变

选择**[Eyedropper]**,修改显示的字体颜色,如图5-16 修改字体颜色。

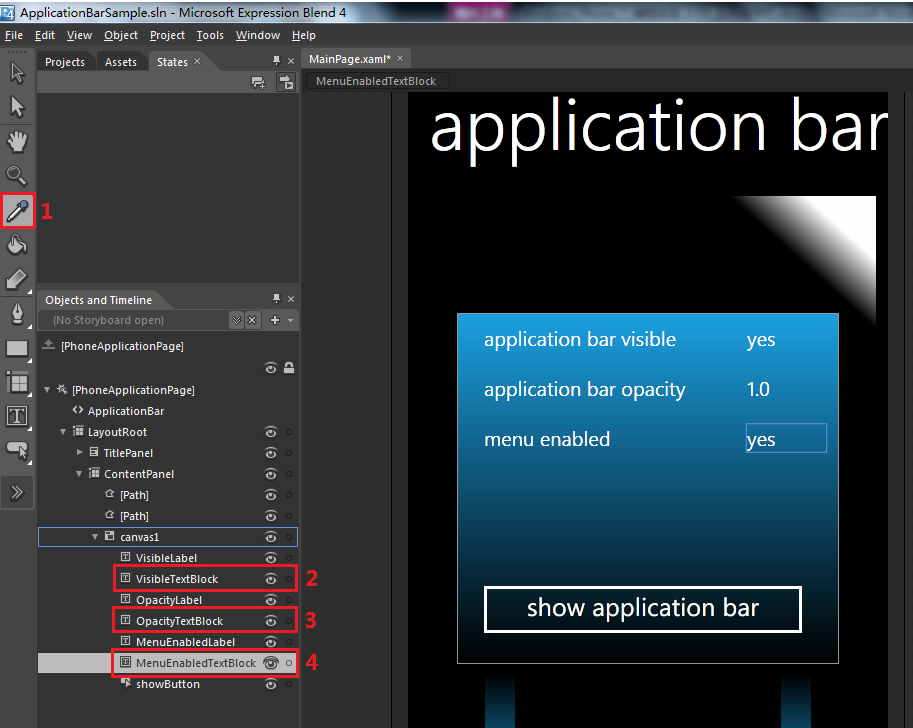


图5-16 修改字体颜色

1. 画面修改三

旋转按钮，在[Objects and Timeline]中选择需要设置的控件”showButton”，在[Properties]属性—[Rotate]旋转，设定[Angle]角度的值为5。如图5-17 旋转按钮。



图5-17 旋转按钮

保存修改。回到Visaul Studio，编译器自动发现了XAML被修改，提示我们是否重新载入，如图5-18 提示信息。点击**[Yes]**。

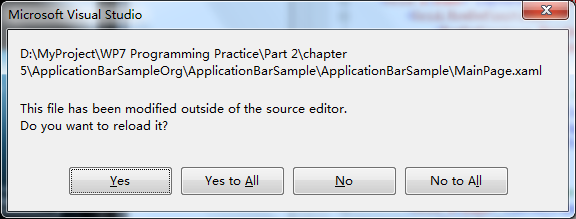


图5-18 提示信息

Expression Blend与Visual Studio的完美无缝结合，实现了界面设计的零代码修改，Expression Blend的更多功能我们在后面的章节中讲述。