Application Bar for Windows Phone

This article explains how to add an application bar to your Windows Phone app. It extends content on MSDN with concrete examples and a downloadable source code example.

23 Sep 2012

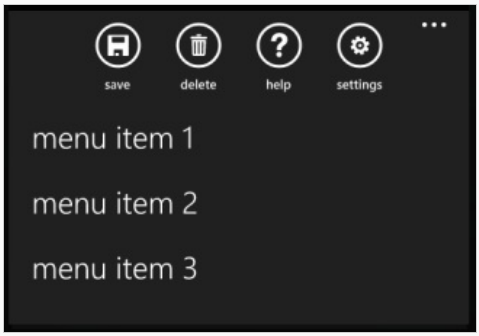
Introduction

XAML

The Application Bar provides access to the Windows Phone app menu. By default the bar contains a set of icon buttons for accessing the most important functionality. Clicking the ellipsis (series of dots) at the top right corner of the bar/panel/window expands it into a list menu (if defined). The application bar also has a minimal mode, which provides only the ellipsis. This is useful for apps where there are no "key" menu options.

This article demonstrates how to create an Application Bar using XAML or with code (C# or VB), how to handle menu item selection, and other features of this resource.

ApplicationBar default size



ApplicationBar expanded with menu. This is displayed when the user taps the ellipsis for view button caption and menu

ApplicationBar mini mode

Application Bar with XAML

First of all you have to open your page in the designer view to see XAML code.

Unless you have a panorama page, you should find the Application Bar code at the end of your XAML code. These lines of code are added by default in the commented form. It will look like this

```
<!--Sample code showing usage of ApplicationBar-->
<!--<phone:PhoneApplicationPage.ApplicationBar>
    …
</phone:PhoneApplicationPage.ApplicationBar>-->
```

Simply uncomment the code by removing the HTML comments ("<!--" and "-->")

In this example we will write from the beginning the XAML code of Application Bar. First of all it must be placed before the last line of XAML, so that it looks like this

```
</phone:PhoneApplicationPage>
```

Your XAML for Application Bar must be placed between these 2 tags

```
<phone:PhoneApplicationPage.ApplicationBar>

</phone:PhoneApplicationPage.ApplicationBar>
```

After opening the tag for the Application Bar you have to set its properties. You can make it with this line of XAML

```
<shell:ApplicationBar IsVisible="True" IsMenuEnabled="True" Opacity="1.0"  Mode="Default">
```

- `IsVisible` - defines whether the application bar is initially visible.
- `IsMenuEnabled` - defines whether the bar has a menu
- `Opacity` - defines whether the underlying app shows through the bar

(C) Copyright Nokia 2013. All rights reserved.

- `Mode` - defines whether the bar is the default (has icons) or is in mini mode.

After that you have to add at least one button (you can also add more). The XAML to add a button is:

```
<shell:ApplicationBarIconButton Click="Save_Click" IconUri="/Images/save.png" Text="save" />
```

- `Click` field is the event which is called when the user clicks on the button
- `Text` sets the label of the button.
- `IconUri` is the Uri of the image of the button. It must be placed in the Image folder (or sub-folder) of Application project folder.

If you don't have an image folder for your buttons you will have to create one. To make it and add new images the procedure is:

1. Go to **Solution Explorer** , right-click your project and click **Add** and then **New Folder**
2. Name the folder **Images**
3. Right-Click the folder "Images", hover the mouse on **Add** and then click **Existing Item**
4. Browse to the location of the icon files, select one or more files, and then click**Add**.
5. In Solution Explorer, select all the new files.
6. In the Properties window, set the property **Build Action** to **content** and **Copy to Output Directory** to **Copy if Newer**. Setting the **Build Action** to **content** is necessary for the image to show up on the application bar.

You can find some default images for your button in these folders: *C:\Program Files\Microsoft SDKs\Windows Phone\v7.1\Icons\dark* or *C:\Program Files (x86)\Microsoft SDKs\Windows Phone\v7.1\Icons\dark*

Adding menu items

You can also add a menu by placing items between these 2 tags:

```
<shell:ApplicationBar.MenuItems>

</shell:ApplicationBar.MenuItems>
```

The XAML for adding a menu item is

```
<shell:ApplicationBarMenuItem Click="Menu1_Click" Text="menu item 1" />
```

Click field is the event which is called when the user clicks on the button

Now your XAML Application Bar is complete. Your XAML should look like this

```
<phone:PhoneApplicationPage.ApplicationBar>
    <shell:ApplicationBar Mode="Default" Opacity="1.0" IsMenuEnabled="True" IsVisible="True">

        <shell:ApplicationBarIconButton Click="Save_Click" IconUri="/Images/save.png" Text="save"/>
        <shell:ApplicationBarIconButton Click="Settings_Click" IconUri="/Images/settings.png" Text="settings"/>

        <shell:ApplicationBar.MenuItems>
            <shell:ApplicationBarMenuItem Click="Menu1_Click" Text="menu item 1" />
            <shell:ApplicationBarMenuItem Click="Menu2_Click" Text="menu item 2" />
        </shell:ApplicationBar.MenuItems>

    </shell:ApplicationBar>
</phone:PhoneApplicationPage.ApplicationBar>
```

Handling events

For handling click events you have to go to the code inside the page class and write a function which handles the event. An example for the Application Bar we created before is:

For C#:

```
private void Save_Click(object sender, EventArgs e)
{
    MessageBox.Show("Save button works!");
    //Do work for your application here.
}

private void Settings_Click(object sender, EventArgs e)
{
    MessageBox.Show("Settings button works!");
    //Do work for your application here.
}

private void Menu1_Click(object sender, EventArgs e)
{
    MessageBox.Show("Menu item 1 works!");
    //Do work for your application here.
}

private void Menu2_Click(object sender, EventArgs e)
{
    MessageBox.Show("Menu item 2 works!");
    //Do work for your application here.
}
```

For VB:

(C) Copyright Nokia 2013. All rights reserved.

```
Private Sub Save_Click(ByVal sender As Object, ByVal e As EventArgs)

    MessageBox.Show("Save button works!")
    'Do work for your application here.
End Sub


Private Sub Settings_Click(ByVal sender As Object, ByVal e As EventArgs)

    MessageBox.Show("Settings button works!")
    'Do work for your application here.
End Sub


Private Sub MenuItem1_Click(ByVal sender As Object, ByVal e As EventArgs)

    MessageBox.Show("Menu item 1 works!")
    'Do work for your application here.
End Sub


Private Sub MenuItem2_Click(ByVal sender As Object, ByVal e As EventArgs)

    MessageBox.Show("Menu item 2 works!")
    'Do work for your application here.
End Sub
```

Your application bar is now complete!

Application Bar using C#/VB

First of all you have to open the code page of your application (usually it is named **MainPage.xaml.cs**, but you can add your Application bar to all your application pages). After that you have to add the statement:

For C#:

```
using Microsoft.Phone.Shell;
```

For VB:

```
Imports Microsoft.Phone.Shell
```

In the page constructor, after the call of `InitializeComponent();` you have to initialize a new ApplicationBar object

```
ApplicationBar = new ApplicationBar();
```

Now you can begin to set all the Application Bar properties you need. In this example all the properties of the Application Bar will be set. You can also choose to set only the ones you need. The code is the same both for C# and VB:

```
 /*This changes the size of the Application Bar when it first appears in the page. /*
ApplicationBar.Mode = ApplicationBarMode.Default;
 /*This changes the opacity of the Application Bar. With "1" the bar is completely opaque, with "0.5" it's partially transparent, wi
ApplicationBar.Opacity = 1.0;
 /*This sets the "visible" property of the Application Bar. On "true" the application bar is visible, on "false" it is hidden*/
ApplicationBar.IsVisible = true;
 /*This property indicates if the user can see menu items when they expand the Application Bar*/
ApplicationBar.IsMenuEnabled = true;
 /*This changes the BackGround color. If you want Default BG Color you can omit it
ApplicationBar.BackgroundColor = Color.FromArgb(120, 255,255, 255);*/
 /*This changes the ForeGround color (visualized on icon labels or menu text). If you want Default FG color you can omit it*/
ApplicationBar.ForeGroundColor = Color.FromArgb(120, 0,0, 0);*/
```

After this you can add buttons or menu items. At least one button is required by the Application Bar, menus are optional.

Begin with adding a `Button`. We have to create an `ApplicationBarIconButton` object first, and then set its icon and label. The icon image of the button must be placed in the Image folder (or sub-folder) of the Application project folder.

If you don't have this folder you have to create it. To make it and add new images the procedure is:

1. Go to **Solution Explorer' *, right-click your project and click*** *Add* and then **New Folder**
2. Name the folder **Images**
3. Right-Click the folder "Images", hover the mouse on **Add** and then click **Existing Item**
4. Browse to the location of the icon files, select one or more files, and then click**Add**.
5. In Solution Explorer, select all the new files.
6. In the Properties window, set the property **Build Action** to **content** and **Copy to Output Directory** to **Copy if Newer**. Setting the **Build Action** to **content** is necessary for the image to show up on the application bar.

You can find some default images for your button in these folders: *C:\Program Files\Microsoft SDKs\Windows Phone\v7.1\Icons\dark* or *C:\Program Files (x86)\Microsoft SDKs\Windows Phone\v7.1\Icons\dark*

Now back to the code

For C#:

```
ApplicationBarIconButton button1 = new ApplicationBarIconButton();
button1.IconUri = new Uri("/Images/YourImage.png", UriKind.Relative);
button1.Text = "button 1";
```


(C) Copyright Nokia 2013. All rights reserved.

```
ApplicationBar.Buttons.Add(button1);
```

For VB:

```
Dim button1 as ApplicationBarIconButton = new ApplicationBarIconButton()
button1.IconUri = new Uri("/Images/YourImage.png", UriKind.Relative)
button1.Text = "button 1"
ApplicationBar.Buttons.Add(button1)
```

You can add more buttons. To add a menu you have to create an 'ApplicationBar'MenuItem *and set its text*

For C#:

```
ApplicationBarMenuItem menuItem1 = new ApplicationBarMenuItem();
menuItem1.Text = "menu item 1";
ApplicationBar.MenuItems.Add(menuItem1);</csharp>

For VB:
<code vb>Dim menuItem1 as ApplicationBarMenuItem = new ApplicationBarMenuItem()
menuItem1.Text = "menu item 1"
ApplicationBar.MenuItems.Add(menuItem1)
```

Add icon buttons and expose event handlers

Now our Application Bar is ready! Time to add icon buttons and menu items expose click events handlers.

For each button and menu item we added in the previous part we have to identify the event which will be called when users click on it.

For C#:

```
button1.Click += new EventHandler(button1_Click);
```

For VB:

```
AddHandler button1.Click, AddressOf button1_Click
```

For the menu:

For C#:

```
menuItem1.Click += new EventHandler(menuItem1_Click);
```

For VB:

```
AddHandler menuItem1.Click, AddressOf menuItem1_Click
```

And then we have to add the call to the specific function for the click input:

For C#:

```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox.Show("Button 1 works!");
    //Do work for your application here.
}

private void menuItem1_Click(object sender, EventArgs e)
{
    MessageBox.Show("Menu item 1 works!");
    //Do work for your application here.
}
```

For VB:

```
Private Sub button1_Click(ByVal sender As Object, ByVal e As EventArgs)

    MessageBox.Show("Button 1 works!")
    'Do work for your application here.
End Sub


Private Sub menuItem1_Click(ByVal sender As Object, ByVal e As EventArgs)

    MessageBox.Show("Menu item 1 works!")
    'Do work for you application here.
End Sub
```

Now your application bar is complete.

Complete code source

Here's an example of complete code:

C#:

```
using Microsoft.Phone.Shell;
```


(C) Copyright Nokia 2013. All rights reserved.

```
public MainPage()
{
    InitializeComponent();

    ApplicationBar = new ApplicationBar();

    ApplicationBar.Mode = ApplicationBarMode.Default;
    ApplicationBar.Opacity = 1.0;
    ApplicationBar.IsVisible = true;
    ApplicationBar.IsMenuEnabled = true;

    ApplicationBarIconButton button1 = new ApplicationBarIconButton();
    button1.IconUri = new Uri("/Images/YourImage.png", UriKind.Relative);
    button1.Text = "button 1";
    ApplicationBar.Buttons.Add(button1);
    button1.Click += new EventHandler(button1_Click);

    ApplicationBarMenuItem menuItem1 = new ApplicationBarMenuItem();
    menuItem1.Text = "menu item 1";
    ApplicationBar.MenuItems.Add(menuItem1);
    menuItem1.Click += new EventHandler(menuItem1_Click);
}

private void button1_Click(object sender, EventArgs e)
{
    MessageBox.Show("Button 1 works!");
    //Do work for your application here.
}

private void menuItem1_Click(object sender, EventArgs e)
{
    MessageBox.Show("Menu item 1 works!");
    //Do work for your application here.
}
```

VB:

```
Imports Microsoft.Phone.Shell

Public Sub New()
    InitializeComponent()

    ApplicationBar = new ApplicationBar()

    ApplicationBar.Mode = ApplicationBarMode.Default
    ApplicationBar.Opacity = 1.0
    ApplicationBar.IsVisible = true
    ApplicationBar.IsMenuEnabled = true

    Dim button1 as ApplicationBarIconButton = new ApplicationBarIconButton()
    button1.IconUri = new Uri("/Images/YourImage.png", UriKind.Relative)
    button1.Text = "button 1"
    ApplicationBar.Buttons.Add(button1)
    AddHandler button1.Click, AddressOf button1_Click

    Dim menuItem1 as ApplicationBarMenuItem = new ApplicationBarMenuItem()
    menuItem1.Text = "menu item 1"
    ApplicationBar.MenuItems.Add(menuItem1)
    AddHandler menuItem1.Click, AddressOf menuItem1_Click
End Sub

Private Sub button1_Click(ByVal sender As Object, ByVal e As EventArgs)

    MessageBox.Show("Button 1 works!")
    'Do work for your application here.
End Sub


Private Sub menuItem1_Click(ByVal sender As Object, ByVal e As EventArgs)

    MessageBox.Show("Menu item 1 works!")
    'Do work for you application here.
End Sub
```


(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.