Windows Phone Mango编程实践

***Windows Phone Mango Programming Practice***

第二篇 Silverlight交互篇

# 启动器和选择器

选择器类似于三十六计中的借刀杀人。应用程序无法直接获得的数据或者任务，利用第三方即通过选择器来获得或者完成，与兵法中的借刀杀人计策原理相通。又有《易经》：“损：损下益上，其道上行。”意指“损”与“益”的转化关系，借用盟友的力量去打击敌人，势必要使盟友受到损失，但盟友的损失正是自己的利益。

## 概述

本章参考和引用Windows Phone官方教程和开发培训包，以及MSDN Windows Phone开发文档。

Windows Phone应用程序模型将每个应用分离成各自独立的沙盒，包括运行时（包括内存的隔离）和文件存储。为了适应需要通用任务的场景，Windows手机公布了一套启动器和选择器的API，允许应用程序间接的访问常用的手机功能。启动器和选择器框架使得Windows Phone应用程序能够向用户提供一套通用的任务，例如打电话，发送电子邮件和拍照片。启动器和选择器的API调用独立的内置应用程序，取代当前运行的应用程序。那么如何理解启动器和选择器呢？

启动器类似于三十六计中的走为上计。启动器调用一个内置的应用程序完成特定的任务，不返回任何数据就轻轻地走了，正如它轻轻的来。PhoneCallTask就是一个例子，应用程序使用启动器调用通话应用程序，并传递电话号码和显示名称等参数。当用户关掉通话应用程序后，调用它的应用程序通常会被重新激活，但是通话应用程序不会返回任何数据或者用户操作结果。即当应用程序调用启动器时，它不会得到返回值。

选择器类似于三十六计中的借刀杀人。应用程序无法直接获得的数据或者任务，利用第三方即通过选择器来获得或者完成，与兵法中的借刀杀人计策原理相通。又有《易经》：“损：损下益上，其道上行。”意指“损”与“益”的转化关系，借用盟友的力量去打击敌人，势必要使盟友受到损失，但盟友的损失正是自己的利益。

选择器调用内置的应用程序完成特定的任务，并向调用它的应用程序返回某种类型的数据。例如，选择器调用“联系人”应用程序，允许你搜索到一个特定联系人。搜索成功之后，联系人信息将被返回。另外一个例子是**PhotoChooserTask**，应用程序可以使用这个选择器来显示选择照片应用程序。用户可以选择照片或取消选择照片。通常发生这种情况时，调用应用程序被重新激活并获取选择器的结果。

注意：在某些情况下，主调应用程序可能永远不会被激活，并且选择器的结果可能永远不会被返回。例如，选取电子邮件地址或者选择照片。选择器启动照片应用程序，此时如果用户可以按下开始按钮（Start），选择另一个应用程序，那么原主调应用程序可能永远不会被激活，并且选择器的结果可能永远不会被返回。

Windows Phone支持的启动器和选择器任务：

**启动器**

|  |  |
| --- | --- |
| **任务** | **描述** |
| EmailComposeTask | 允许应用程序启动电子邮件应用程序并创建一条新消息；以此来让用户从应用程序发送电子邮件。 |
| MarketplaceDetailTask | 允许应用程序启动Windows Phone Market客户端应用程序并显示指定产品的详细信息页面。 |
| MarketplaceHubTask | 允许应用程序启动WindowsPhone Market的客户端应用程序。 |
| MarketplaceReviewTask | 允许应用程序启动Windows Phone Market客户端应用程序并显示指定产品的评论信息页面。 |
| MarketplaceSearchTask | 允许应用程序启动Windows Phone Market客户端应用程序并显示指定搜索条件的检索结果。 |
| MediaPlayerLauncher | 允许应用程序启动媒体播放器。 |
| PhoneCallTask | 允许应用程序启动电话应用程序；使得用户能够在应用程序中开始打电话。 |
| SaveEmailAddressTask | 允许应用程序启动联系人应用程序；以此允许用户从应用程序中保存电子邮件地址到一个新的或现有的联系人。 |
| SavePhoneNumberTask | 允许应用程序启动联系人应用程序；以此允许用户从应用程序中保存电话号码到一个新的或现有的联系人。 |
| SearchTask | 允许应用程序启动Web 搜索应用程序。 |
| SmsComposeTask | 允许应用程序启动SMS应用程序。 |
| WebBrowserTask | 允许应用程序启动Web浏览器应用程序。 |

表12-1 启动器

**选择器**

|  |  |
| --- | --- |
| **任务** | **描述** |
| EmailAddressChooserTask | 允许应用程序启动联系人应用程序；使用它来获取用户选定的联系人的电子邮件地址。 |
| CaptureCameraTask | 允许应用程序启动照相机应用程序；使用户能够从你的应用中拍照片。 |
| PhoneNumberChooserTask | 允许应用程序启动联系人应用程序；使用它来获取用户选定的联系人的电话号码 |
| PhotoChooserTask | 允许应用程序启动照片选择应用程序；使用它来让用户选择照片。 |

表12-2 选择器

启动器和选择器的使用要注意应用程序生命周期，关于应用程序生命周期的讲述，请参考下一章节的介绍。

下面的任务被调用时，应用程序不会自动触发逻辑删除机制：

PhotoChooserTask PhotoChooserTask

CameraCaptureTask CameraCaptureTask

MediaPlayerLauncher MediaPlayerLauncher

EmailAddressChooserTask EmailAddressChooserTask

PhoneNumberChooserTask PhoneNumberChooserTask

在以下的情况下，后台的应用程序将立即被执行逻辑删除 ：

* 用户启动另一个应用程序的导航，例如，用户按下启动键；
* 应用程序调用的启动器和选择器不在上面的列表中；
* 系统需要更多的资源来执行前台活动。

## Windows Phone模拟器对启动器和选择器的支持

下表显示了Windows Phone模拟器对于启动器和选择器的支持。

|  |  |
| --- | --- |
| **类名称** | **Windows Phone模拟器** |
| CameraCaptureTask | 因为Windows Phone模拟器没有相机。，所以Windows Phone模拟器返回一个默认的图像应用程序。 |
| EmailAddressChooserTask | 目前与Windows Phone真机没有差异。 |
| EmailComposeTask | 这不是在Windows Phone模拟器的功能，也不能在Windows Phone模拟器中创建电子邮件账户。 |
| MarketplaceDetailTask | 目前与Windows Phone真机没有差异。 |
| MarketplaceHubTask | 目前与Windows Phone真机没有差异。 |
| MarketplaceReviewTask | 因为没有Windows Live ID的缘故，所以打开产品的打分和评论页面时会显示错误。 |
| MarketplaceSearchTask | 目前与Windows Phone真机没有差异。 |
| MediaPlayerLauncher | 应用程序可以启动媒体播放器，用户可以播放音乐，但视频不显示。 |
| PhoneCallTask | Windows Phone模拟器使用假GSM和总有虚假的SIM卡，电话连接状态是通过模拟器模拟的。 |
| PhoneNumberChooserTask | 目前与Windows Phone真机没有差异。 |
| PhotoChooserTask | 目前与Windows Phone真机没有差异。 |
| SaveEmailAddressTask | 目前与Windows Phone真机没有差异。 |
| SavePhoneNumberTask | 目前与Windows Phone真机没有差异。 |
| SearchTask SearchTask | 目前与Windows Phone真机没有差异。 |
| SmsComposeTask | Windows Phone模拟器启动电话号码和填充消息SMS客户端，不反悔任何状态或错误返回到应用程序，且总是出现在SMS消息发送成功。 |
| WebBrowserTask | 目前与Windows Phone真机没有差异。 |

表12-3 Windows Phone模拟器对于启动器和选择器的支持

## 动手实践——启动器和选择器的应用

本例中使用全景控件的MVVM模式展示启动器（Launcher）和选择器（chooser）调用任务的功能，如图12-1 启动器和选择器。

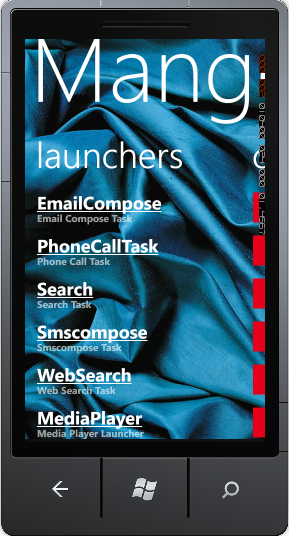
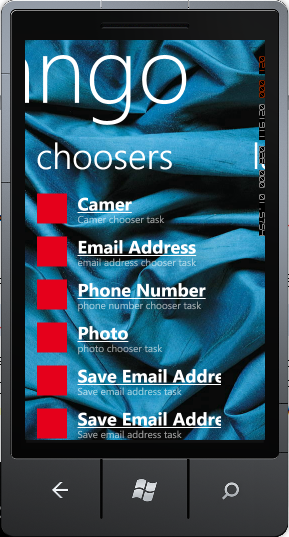
 

图12-1 启动器和选择器。

1. 全景控件的MVVM模式

从**开始 | 所有程序 | Microsoft Visual Studio 2010 Express | Microsoft Visual Studio 2010 Express for Windows Phone**中打开 Microsoft Visual Studio 2010 Express for Windows Phone。或者从**开始 | 所有程序| Microsoft Visual Studio 2010打开**Visual Studio 2010。

在**File**菜单中，选择**Project/Solution**，导航到**Part 2\chapter 12\LaunchersChoosers\_Begin**文件夹，选择**LaunchersChoosers.sln**，然后点击**OK**。

本例中全景控件采用Model-View-ViewModel (MVVM)模式绑定为设计模式数据和运行模式数据。设计模式的数据定义在**DesignData\MainViewModelSampleData.xaml**，运行模式定义在**ViewModels\MainViewModels.cs**中的**MainViewModel**类的**LoadData**()。

设计模式的DataContext定义方法，在MainPage.xaml文件的头部定义设计模式下显示的数据源文件。

Silverlight Project: LaunchersChoosers File: MainPage.xaml

d:DataContext="{d:DesignData DesignData/MainViewModelSampleData.xaml}"

运行模式下**MainViewModel**类的数据加载代码。

Silverlight Project: LaunchersChoosers File: ViewModels\MainViewModels.cs

/// <summary>

/// Creates and adds a few ItemViewModel objects into the Items collection.

/// </summary>

public void LoadData()

{

this.Items.Add(new ItemViewModel() { LineOne="EmailCompose", LineTwo="Email Compose Task", LineThree="/Launchers/EmailCompose.xaml" });

this.Items.Add(new ItemViewModel() { LineOne="PhoneCallTask", LineTwo="Phone Call Task", LineThree="/Launchers//PhoneCall.xaml" });

this.Items.Add(new ItemViewModel() { LineOne="Search", LineTwo="Search Task", LineThree="/Launchers//Search.xaml" });

this.Items.Add(new ItemViewModel() { LineOne="Smscompose", LineTwo="Smscompose Task", LineThree="/Launchers/Smscompose.xaml" });

this.Items.Add(new ItemViewModel() { LineOne="WebBrowser", LineTwo="Web Browser Task", LineThree="/Launchers/WebBrowser.xaml" });

this.Items.Add(new ItemViewModel() { LineOne="MediaPlayer", LineTwo="Media Player Launcher", LineThree="/Launchers/MediaLauncher.xaml" });

this.Items.Add(new ItemViewModel() { LineOne="Marketplace", LineTwo="MarketPlaceDetail Task and MarketplaceHub Task", LineThree="/Launchers/MarketPlace.xaml" });

this.ChoosersItems.Add(new ItemViewModel() { ChoosersLineOne = "Camer", ChoosersLineTwo = "Camer chooser task", ChoosersLineThree = "/Choosers/CamerChooser.xaml" });

this.ChoosersItems.Add(new ItemViewModel() { ChoosersLineOne="Email Address", ChoosersLineTwo="email address chooser task", ChoosersLineThree="/Choosers/EmailAddressChooser.xaml" });

this.ChoosersItems.Add(new ItemViewModel() { ChoosersLineOne="Phone Number", ChoosersLineTwo="phone number chooser task", ChoosersLineThree="/Choosers//PhoneNumberChooser.xaml" });

this.ChoosersItems.Add(new ItemViewModel() { ChoosersLineOne="Photo", ChoosersLineTwo="photo chooser task", ChoosersLineThree="/Choosers/PhotoChooser.xaml" });

this.ChoosersItems.Add(new ItemViewModel() { ChoosersLineOne="Save Email Address", ChoosersLineTwo="Save email address task", ChoosersLineThree="/Choosers/SaveEmailAddress.xaml" });

this.ChoosersItems.Add(new ItemViewModel() { ChoosersLineOne="Save Email Address", ChoosersLineTwo="Save email address task", ChoosersLineThree="/Choosers/SaveEmailAddress.xaml" });

this.ChoosersItems.Add(new ItemViewModel() { ChoosersLineOne="Save Phone number", ChoosersLineTwo="Save Phone number task", ChoosersLineThree="/Choosers/SavePhoneNumber.xaml" });

this.IsDataLoaded = true;

}

全景控件的Launchers选项将ItemsSource绑定为MainViewModel类的 Items，Choosers选项将将ItemsSource绑定为MainViewModel类的Choosers Items。Launchers选项和Choosers选项都在ListBox控件中采用数据绑定的方式显示HyperlinkButton的内容和链接地址的数据源。修改全景控件的代码如下所示。

Silverlight Project: LaunchersChoosers File: MainPage.xaml

<!--Panorama control-->

<controls:Panorama Title="Mango">

<controls:Panorama.Background>

<ImageBrush ImageSource="PanoramaBackground.png"/>

</controls:Panorama.Background>

<!--Panorama item one-->

<controls:PanoramaItem Header="launchers">

<ListBox Margin="0,0,-12,0" ItemsSource="{Binding Items}" FontWeight="Black">

<ListBox.ItemTemplate>

<DataTemplate>

<StackPanel Margin="0,0,0,17" Width="432">

<HyperlinkButton

Content="{Binding LineOne}"

HorizontalAlignment="Left"

HorizontalContentAlignment="Left"

NavigateUri="{Binding LineThree}"

FontSize="36" FontWeight="Black">

</HyperlinkButton>

<TextBlock Text="{Binding LineTwo}" TextWrapping="Wrap" Margin="12,-6,12,0" Style="{StaticResource PhoneTextSubtleStyle}"/>

</StackPanel>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

</controls:PanoramaItem>

<!--Panorama item two-->

<controls:PanoramaItem Header="choosers">

<ListBox Margin="0,0,-12,0" ItemsSource="{Binding ChoosersItems}">

<ListBox.ItemTemplate>

<DataTemplate>

<StackPanel Orientation="Horizontal" Margin="0,0,0,17">

<Rectangle Height="60" Width="60" Fill="#FFE5001b" Margin="12,0,9,0"/>

<StackPanel Width="311">

<HyperlinkButton

Content="{Binding ChoosersLineOne}"

HorizontalAlignment="Left"

HorizontalContentAlignment="Left"

NavigateUri="{Binding ChoosersLineThree}"

FontSize="36" FontWeight="Black">

</HyperlinkButton>

<TextBlock Text="{Binding ChoosersLineTwo}" TextWrapping="Wrap" Margin="12,-6,12,0" Style="{StaticResource PhoneTextSubtleStyle}"/>

</StackPanel>

</StackPanel>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

</controls:PanoramaItem>

</controls:Panorama>

1. 添加调用启动器任务的代码文件

在 LaunchersChoosers 工程中，加入启动器调用任务的代码文件。右键点击Launchers文件夹，选择“ Add ”—“ Existing Item… ”，加入Launchers文件夹下的所有文件。如图12-2 添加代码文件

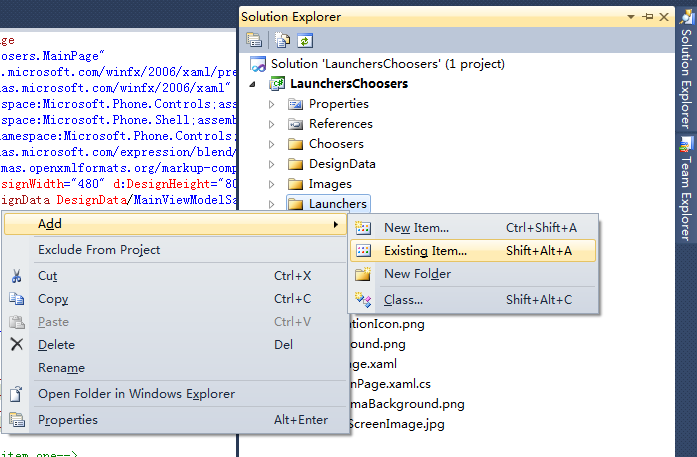
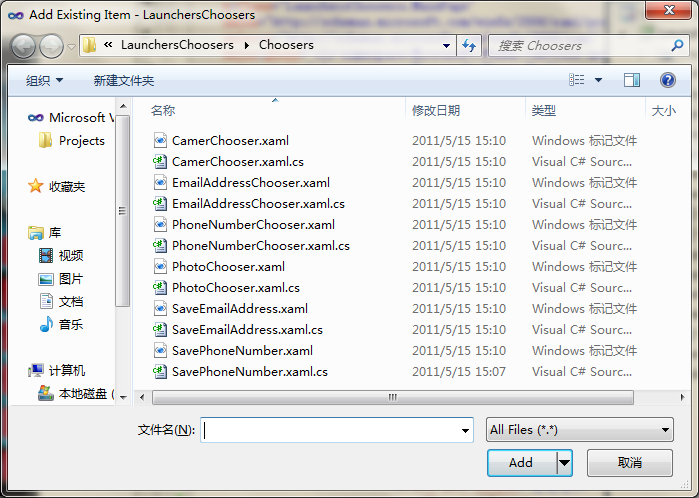
 

图12-2 添加代码文件

1. 添加调用选择器的代码文件

选择器调用内置的应用程序完成特定的任务，并向调用它的应用程序返回某种类型的数据。例如说一张图片、一个联络人信息等等。要使用 Chooser 要先引入 Microsoft.Phone.Task 的命名空间。

在 LaunchersChoosers 工程中，加入启动器调用任务的代码文件。右键点击Choosers文件夹，选择“ Add ”—“ Existing Item… ”，加入Choosers文件夹下的所有代码文件。如图12-13 添加代码文件。

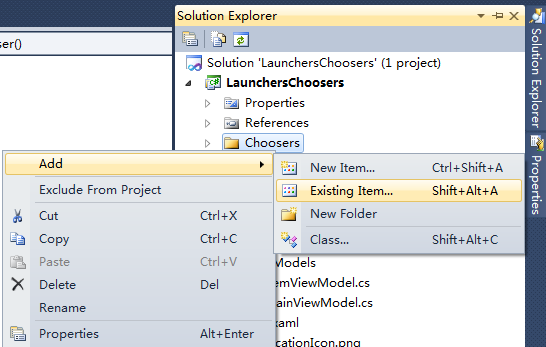
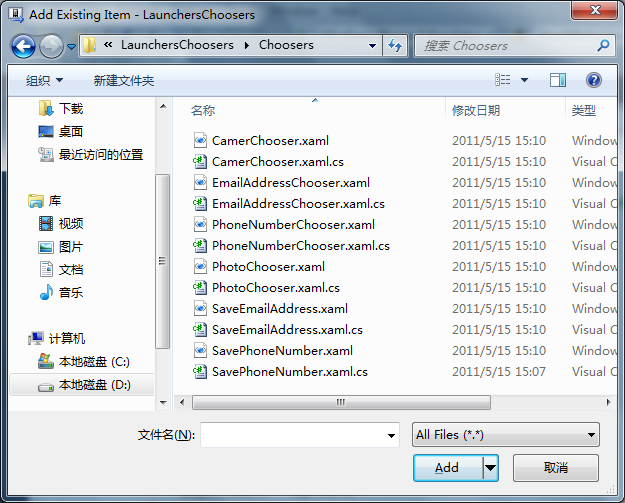
 

图12-3 添加代码文件

### 启动器之EmailComposeTask

EmailComposeTask 调用系统默认的Email发送功能，允许设定收件人和邮件正文等内容。笔者做了以下的接口，在模拟器执行的时候会出现右图的错误提示，这是因为内建的开发用仿真器没有设定Email相关的账号，因此无法做寄送Email的动作。您可以将应用程序部署到真实的Windows Phone手机上做实际的测试。

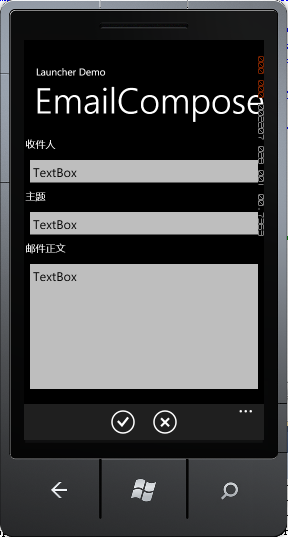
 

图12-4 EmailComposeTask

从EmailCompose.xaml中读取Email收件人、邮件主题和邮件正文内容，设置EmailComposeTask的To、Subject和Body属性。调用Show方法。

Silverlight Project: LaunchersChoosers File: Launchers/EmailCompose.xaml.cs

EmailComposeTask ect = new EmailComposeTask();

ect.To = txtEmailAddress.Text;

ect.Subject = txtSubject.Text;

ect.Body = txtMailBody.Text;

ect.Show();

下表是EmailComposeTask的属性介绍。

|  |  |
| --- | --- |
| **属性** | **描述** |
| Body | 邮件正文 |
| Cc | 抄送地址 |
| Subject | 主题 |
| To | 收件人 |

表12-4 属性介绍

**注意**

在输入Email的时候可以搭配选择器EmailAddressChooserTask来使用会更加人性化。

### 启动器之PhoneCallTask

PhoneCallTask 是能够让您在应用程序中去执行拨打电话的功能。执行 PhoneCallTask 时，需要先指定电话号码以及显示在画面上的名称 (DisplayName)，之后呼叫 Show 的方法；呼叫 Show 方法之后，首先会请使用者确认是否要拨打电话之后才会会进行拨打电话的动作。如果在通话过程中切换了页面，如下面最右侧的图所示，将在Windows Phone屏幕的最上端显示通过事件，轻触其就可展开通话的页面。

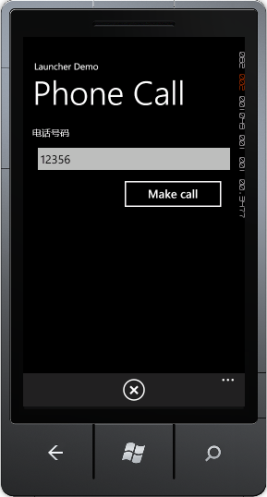
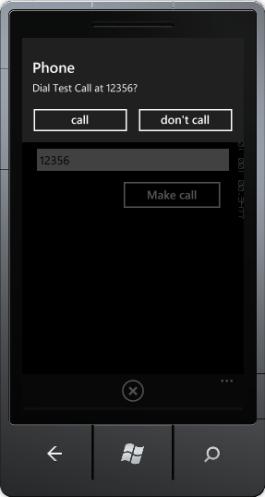
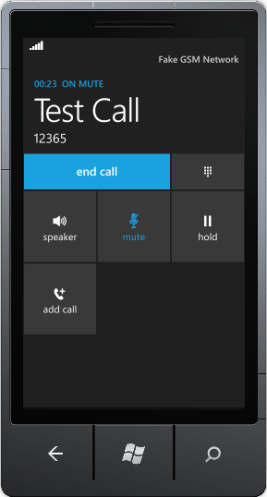
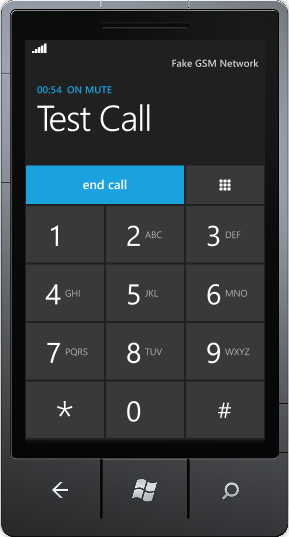
    

图12-5 PhoneCallTask

实现调用启动器PhoneCallTask任务执行电话呼叫的代码如下，其中电话呼叫的号码为文本框txtPhoneNo中输入的号码。

Silverlight Project: LaunchersChoosers File: Launchers/PhoneCall.xaml.cs

PhoneCallTask pct = new PhoneCallTask();

pct.DisplayName = "Test Call";

pct.PhoneNumber = txtPhoneNo.Text;

pct.Show();

### 启动器之SerachTask

SearchTask 能够让应用程序指定查询的关键词，并且去启动系统默认的查询功能。当第一次去启动 SearchTask时，系统会询问使用者是否允许装置利用 GPS/AGPS 等方式去取得目前所在位置的一些相关信息，确认之后显示搜索结果。

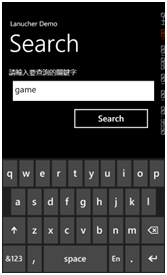
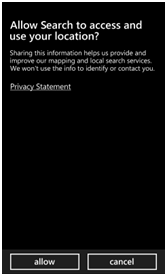
  

图12-6 SearchTask

Silverlight Project: LaunchersChoosers File: Launchers/Search.xaml.cs

SearchTask searchTask = new SearchTask();

searchTask.SearchQuery = txtInput.Text;

searchTask.Show();

### 启动器之SmscomposeTask

SmscomposeTask允许应用程序呼叫系统的短信发送功能。使用的方式跟 EmailComposeTask相似，只要设定接收端的号码以及 SMS 内容之后，就可以启动系统默认的短信发送接口。

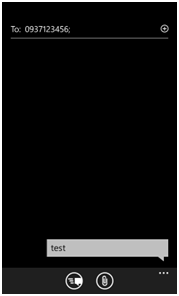
 

图12-7 短信发送

添加Microsoft.Phone.Tasks 的引用。

Silverlight Project: LaunchersChoosers File: Launchers/Smscompose.xaml.cs

using Microsoft.Phone.Tasks;

SmscomposeTask调用代码如下。

Silverlight Project: LaunchersChoosers File: Launchers/Smscompose.xaml.cs

SmsComposeTask sct = new SmsComposeTask();

sct.To = txtPhoneNo.Text;

sct.Body = txtMessage.Text;

sct.Show();

### 启动器之Web Search Task

WebBrowserTask具有启动浏览器的功能。

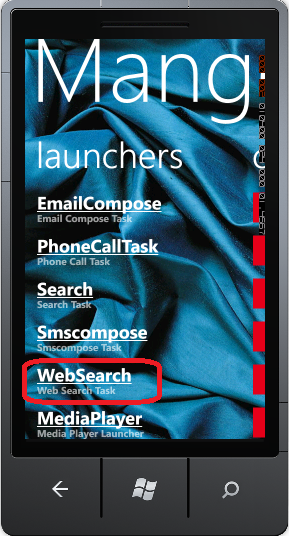
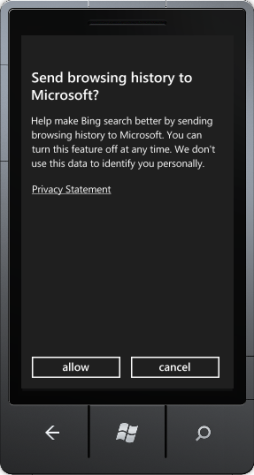
   

图12-8 WebSearch

添加引用如下。

Silverlight Project: LaunchersChoosers File: Launchers/WebBrowser.xaml.cs

using Microsoft.Phone.Tasks;

using System.Diagnostics;

using System.Windows.Navigation;

读取TextBox的关键词，指定WebBrowserTask的Uri为Bing搜索，调用WebBrowserTask的Show方法显示搜索的结果，代码如下。

Silverlight Project: LaunchersChoosers File: Launchers/WebBrowser.xaml.cs

WebBrowserTask webBrowserTask = new WebBrowserTask();

const string url = "http://m.bing.com/search?q={0}&a=results";

webBrowserTask.Uri = new Uri(string.Format(url, txtInput.Text));

webBrowserTask.Show();

在页面的OnNavigatedFrom事件中处理页面瞬态数据，将页面瞬态数据保存至PhoneApplicationPage.State中；在OnNavigatedTo事件中从PhoneApplicationPage.State中恢复页面瞬态数据。

const string inputStateKey = "input";

protected override void OnNavigatedFrom(NavigationEventArgs e)

{

Debug.WriteLine("\*\*\*\t In OnNavigatedFrom function of WebSearchPage\t \*\*\*");

if (State.ContainsKey(inputStateKey))

State.Remove(inputStateKey);

State.Add(inputStateKey, txtInput.Text);

base.OnNavigatedFrom(e);

}

protected override void OnNavigatedTo(NavigationEventArgs e)

{

Debug.WriteLine("\*\*\*\t In OnNavigatedTo function of WebSearchPage\t \*\*\*");

if (State.ContainsKey(inputStateKey))

{

string input = (string)State[inputStateKey];

txtInput.Text = input;

State.Remove(inputStateKey);

}

base.OnNavigatedTo(e);

}

### 启动器之MediaPlayerLanucher

MediaPlayerLanucher顾名思义就是启动MediaPlayer播放影音视频。MediaPlayerLauncher启动应用程序和媒体播放器播放指定的媒体文件。媒体文件可以是存储在独立存储空间中的，也可以是绑定在应用程序中的，因此在播放时应使用MediaLocationType枚举类型指定媒体文件存放的位置。MediaPlayerLanucher允许指定一个或多个控件媒体播放器的控件，指定方法是使用按位或的方式设定MediaPlaybackControls值。

|  |  |
| --- | --- |
| **属性** | **说明** |
| Controls | 设置应用程序调用的播放器控件。 |
| Location | 媒体文件的存储位置MediaLocationType枚举用于指定存储位置为独立存储空间或应用程序的安装文件夹。  MediaLocationType.Install表示文件位于应用程序安装文件夹中；  MediaLocationType.Data表示文件位于独立存储空间中。 |
| Media | 以Uri的方式来表示文件 |

表12-5 MediaPlayerLanucher

添加Media文件夹下的影音文件mymovie.wmv。右键点击“Media”文件夹，选择“Add”—“Existing Item…”。如图12-8 添加Media文件。

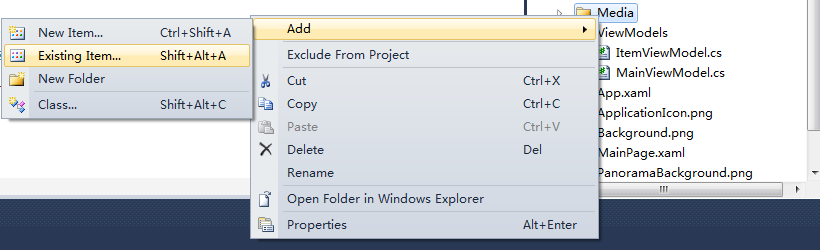


图12-9 添加Media文件

设置影音文件mymovie.wmv的属性，右键点击影音文件mymovie.wmv，选择属性“Properties”。如图12-9 选择属性。

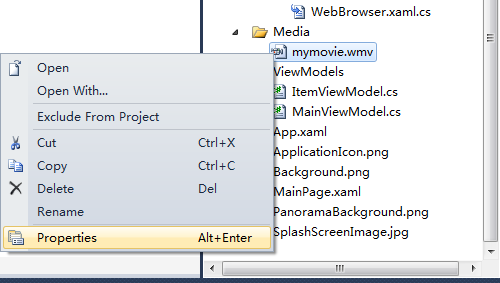


图12-10 选择属性

将“Build Action”属性设定成“Content”，“copy to output directory”属性设定为“copy if newer”。

打开Launchers/MediaLauncher.xaml中，我们设置了一个CheckBox来选择播放影音文件的源。在运行时选择CheckBox则播放Channel9网络视频文件WPMetroDesignOverivew，否则播放应用程序中的视频文件mymovie.wmv。

Silverlight Project: LaunchersChoosers File: Launchers/ MediaLauncher.xaml

<StackPanel Grid.Row="0" Orientation="Horizontal">

<CheckBox x:Name="chkUseExternalMedia"

Content="Channel9 WPMetroDesignOverivew"/>

</StackPanel>

<StackPanel Grid.Row="2" Orientation="Vertical">

<Button Content="Play Video" x:Name="btnPlayVideo" Click="btnPlayVideo\_Click"/>

</StackPanel>

MediaPlayerLauncher的Location属性设置为Install，则播放应用程序安装文件夹中的影音文件。

Silverlight Project: LaunchersChoosers File: Launchers/MediaLauncher.xaml.cs

MediaPlayerLauncher mediaPlayerLauncher = new MediaPlayerLauncher();

if (chkUseExternalMedia.IsChecked.Value)

{

mediaPlayerLauncher.Media = new Uri("http://ecn.channel9.msdn.com/o9/ch9/4/1/9/7/4/5/WPMetroDesignOverivew\_ch9.wmv", UriKind.Absolute);

}

else

{

//means is a resource of the app, otherwise it will try to resolve it in Data (IsolatedStorage) for application

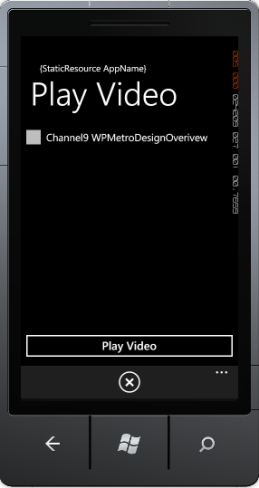
mediaPlayerLauncher.Location = MediaLocationType.Install;

mediaPlayerLauncher.Media = new Uri("Media/mymovie.wmv", UriKind.Relative);

}

mediaPlayerLauncher.Show();

运行效果如图 播放影音文件。

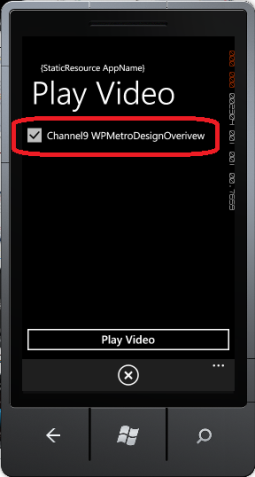
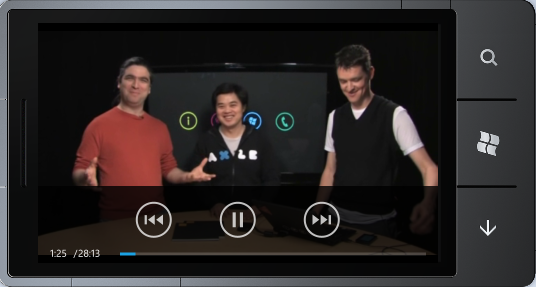
 

图12-11 播放影音文件

### 启动器之MarketPlaceDetailTask

MarketPlaceDetailTask启动系统内建的 MarketPlace 应用程序，并且可以指定要浏览的应用程序 ID。MarketPlaceDetailTask的重要属性如下表12-6 MarketPlaceDetailTask属性所示。

|  |  |
| --- | --- |
| **属性** | **说明** |
| ContentIdentifier | 指定应用程序 ID (是一个 GUID 值)，如果没有指定 (也就是 null) 的话，便会以目前执行的应用程序为目标 |
| ContentType | 指定应用程序的类型，目前只能指定为Applications。 |

表12-6 MarketPlaceDetailTask属性

Silverlight Project: LaunchersChoosers File: Launchers/MarketPlace.xaml.cs

MarketplaceDetailTask mdt = new MarketplaceDetailTask();

//ContentType设定为Applcation

mdt.ContentType = MarketplaceContentType.Applications;

mdt.ContentIdentifier = "Application ID";

mdt.Show();

### 启动器之MarketplaceHubTask

MarketlaceHubTask 主要的功用是启动后便会带领使用者直接联机到 Marketplace,要注意的是 ContentType 属性，可以设定为 Application 与 Music。

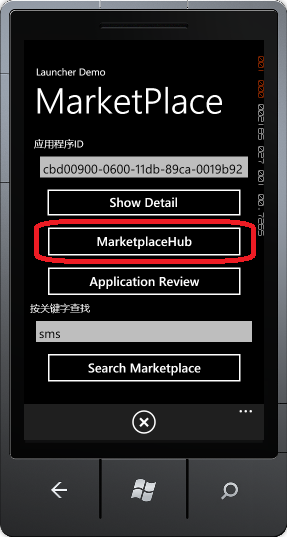
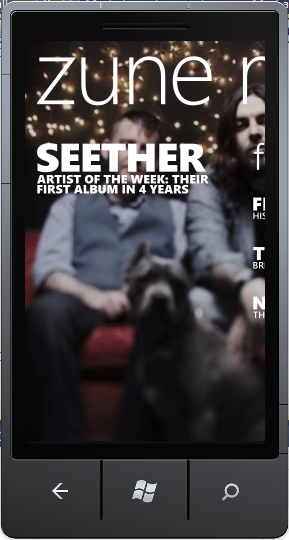
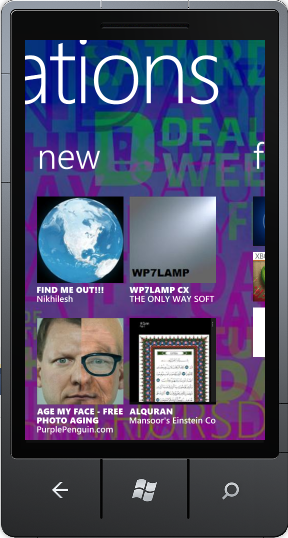
  

图12-12 MarketPlaceHubTask

上图中中间画面是当 ContentType 设定为 Music 时的画面，最右边则是设定为 Application 时。

Silverlight Project: LaunchersChoosers File: Launchers/MarketPlace.xaml.cs

MarketplaceHubTask mht = new MarketplaceHubTask();

//mht.ContentType = MarketplaceContentType.Applications;

mht.ContentType = MarketplaceContentType.Music;

mht.Show();

### 启动器之MarketplaceReviewTask

MarketplcaeReviewTask 的用途是在启动之后会连到 Marketplace 的页面，并直接的为您的应用程序做评分、建议等地动作。

Silverlight Project: LaunchersChoosers File: Launchers/MarketPlace.xaml.cs

MarketplaceReviewTask mrt = new MarketplaceReviewTask();

mrt.Show();

### 启动器之MarketPlaceSearchTask

MarketplaceSearchTask 可以让您搜寻Marketplace上的应用程序或是音乐 (通过设定 ContentType 的属性)，另外 SearchTerms 可以指定关键词。

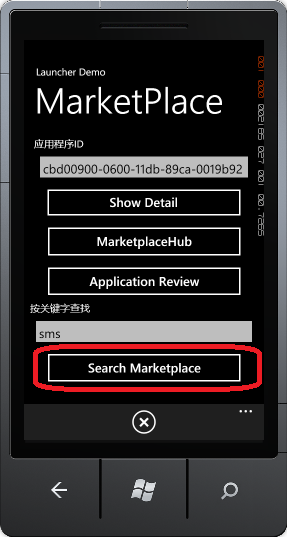
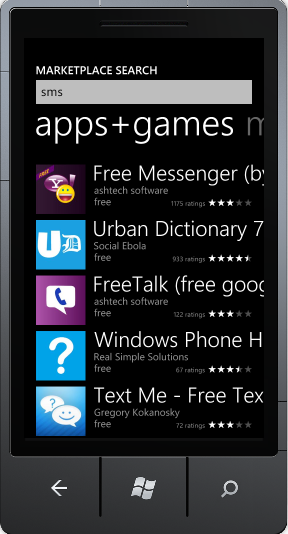
 

图12-12 Marketplace

Silverlight Project: LaunchersChoosers File: Launchers/MarketPlace.xaml.cs

MarketplaceSearchTask mst = new MarketplaceSearchTask();

//可以将ContentType属性设定为Muisc，也可以是Applications

//mst.ContentType = MarketplaceContentType.Music;

mst.ContentType = MarketplaceContentType.Applications;

mst.SearchTerms = txtSearchTerms.Text;

mst.Show();

### 选择器之CameraCaptureTask

CameraCaptureTask调用Windows Phone的拍照任务，下图右上角的 http://i.msdn.microsoft.com/gg454714.windowsphone10_11(zh-tw,MSDN.10).png图案就是拍照的按钮。

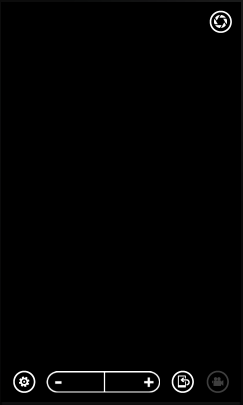


图12-14 CameraCaptureTask

引入Microsoft.Phone.Tasks的命名空间。

Silverlight Project: LaunchersChoosers File: Choosers/ CamerChooser.xaml.cs

using Microsoft.Phone.Tasks;

using System.Windows.Media.Imaging;

初始化时加载拍照事件处理函数。

Silverlight Project: LaunchersChoosers File: Choosers/ CamerChooser.xaml.cs

CameraCaptureTask cct ;

public CamerChooser()

{

InitializeComponent();

//建议在初始化完成之后就加载事件处理函数,这与application lift cycle有关

cct = new CameraCaptureTask();

cct.Completed += new EventHandler<PhotoResult>(cct\_Completed);

}

在CamerChooser.xaml中的Image控件显示拍照后的图片，Button控件的Click事件处理函数中调用CameraCaptureTask。

Silverlight Project: LaunchersChoosers File: Choosers/ CamerChooser.xaml

<Grid x:Name="ContentGrid" Grid.Row="1">

<Image Height="489" HorizontalAlignment="Left" Margin="21,19,0,0" Name="image1" Stretch="Fill" VerticalAlignment="Top" Width="447" />

<Button Content="Take a photo" Height="82" HorizontalAlignment="Left" Margin="191,535,0,0" Name="btnShot" VerticalAlignment="Top" Width="283" Click="btnShot\_Click" />

</Grid>

Button控件的Click事件处理函数btnShot\_Click中通过CameraCaptureTask的Show方法调用选择器。

Silverlight Project: LaunchersChoosers File: Choosers/ CamerChooser.xaml.cs

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

{

//呼叫Chooser

cct.Show();

}

在选择器的使用上，最主要的就是处理拍照的Completed事件：将照片显示在Image控件中。

Silverlight Project: LaunchersChoosers File: Choosers/ CamerChooser.xaml.cs

void cct\_Completed(object sender, PhotoResult e)

{

//判定结果是否成功

if (e.TaskResult == TaskResult.OK)

{

BitmapImage bmpSource = new BitmapImage();

bmpSource.SetSource(e.ChosenPhoto);

image1.Source = bmpSource;

}

else

{

image1.Source = null;

}

}

在处理 Completed 的事件当中，必须要先判断 TaskResult 属性，在这个属性当中，可以取得拍照动作的结果，例如当使用者按下确定 (Accept) 的按钮时，会响应 OK，而如果使用者按下返回键呢？这时候回传的就会是 Cancel 的状态了。

而怎么取得拍摄的照片呢？主要便是利用**PhotoResult**事件的**ChoosenPhoto**属性，**ChoosenPhoto**属性指向实体照片位置的数据流。拍照后的图片是不会直接的储存到应用程序所属的隔离储存空间中的，因为不同应用程序之间是不能去交叉存取隔离储存空间中的文件。

### 选择器之EmailAddressChooserTask

EmailAddressChooserTask主要是用来取得联络人的 Email 数据，此任务可与EmailComposeTask结合使用完成邮件客户端应用程序的功能。

与CameraCaptureTask调用方法相似。在初始化时加载EmailAddressChooserTask的Completed事件处理函数。

Silverlight Project: LaunchersChoosers File: Choosers/ EmailAddressChooser.xaml.cs

EmailAddressChooserTask eac;

public EmailAddressChooser()

{

InitializeComponent();

eac = new EmailAddressChooserTask();

eac.Completed += new EventHandler<EmailResult>(eac\_Completed);

}

使用EmailAddressChooserTask的Show方法调用选择器。

Silverlight Project: LaunchersChoosers File: Choosers/ EmailAddressChooser.xaml.cs

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

{

eac.Show();

}

EmailAddressChooserTask的Completed事件处理函数eac\_Completed。将选择器返回的Email信息赋值给TextBox控件显示。

Silverlight Project: LaunchersChoosers File: Choosers/ EmailAddressChooser.xaml.cs

void eac\_Completed(object sender, EmailResult e)

{

if (e.TaskResult == TaskResult.OK)

{

textBox1.Text = e.Email;

}

}

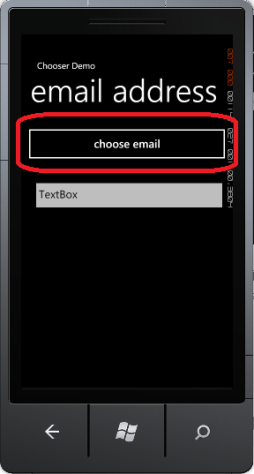
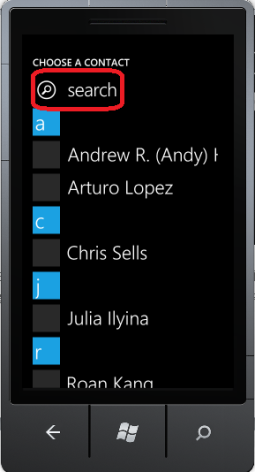
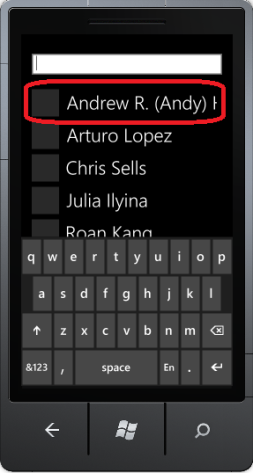
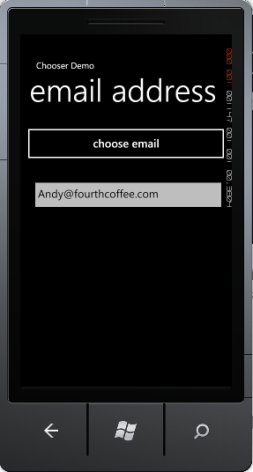
   

图12-15 EmailAddressChooserTask

### 选择器之PhoneNumberChooserTask

PhoneNumberChooserTask主要是用来选择联络人的电话号码。

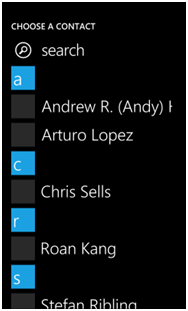
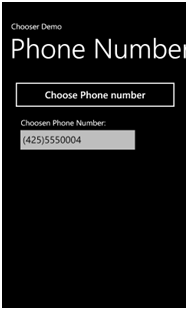
 

图12-16 PhoneNumberChooserTask

Silverlight Project: LaunchersChoosers File: Choosers/ PhoneNumberChooser.xaml.cs

PhoneNumberChooserTask pnc;

public PhoneNumberChooser()

{

InitializeComponent();

pnc = new PhoneNumberChooserTask();

pnc.Completed += new EventHandler<PhoneNumberResult>(pnc\_Completed);

}

使用PhoneNumberChooserTask的Show方法调用选择器。

Silverlight Project: LaunchersChoosers File: Choosers/ PhoneNumberChooser.xaml.cs

private void button1\_Click(object sender, RoutedEventArgs e)

{

pnc.Show();

}

PhoneNumberChooserTask的Completed事件处理函数pnc\_Completed。将选择器返回的Email信息赋值给TextBox控件显示。

Silverlight Project: LaunchersChoosers File: Choosers/ EmailAddressChooser.xaml.cs

void pnc\_Completed(object sender, PhoneNumberResult e)

{

if (e.TaskResult == TaskResult.OK)

{

textBox1.Text = e.PhoneNumber;

}

}

### 选择器之PhotoChooserTask

PhotoChooserTask是用来选择图片。PhotoChooser具有 ShowCamera 的属性，ShowCamera 的属性是一个boolean型态，当设定为 true 时，在选择图片的画面下方中，会出现拍照的按钮，让用户将相机拍照的图片作为数据源。

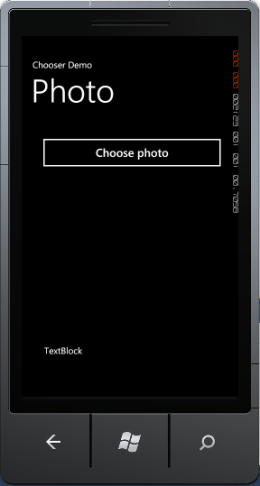
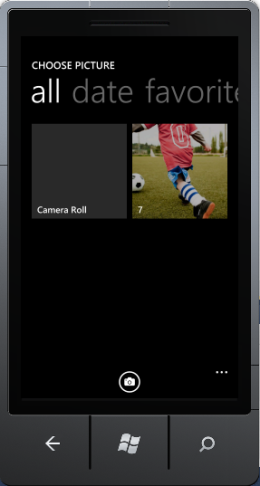
 

图12-17 PhotoChooserTask

通过设置PhotoChooser的PixelHeight和PixelWidth的属性，可以实现对原始图像裁剪的功能。比如说，现在应用程序要让用户设定大头贴，大头贴的尺寸只需要 100\*10，此时需要处理图片的尺寸。通过设定PixelHeight和PixelWidth的属性之后，当用户选定照片会出现裁切的矩形方框，矩形方框会依照您设定的长宽比例自动调整。

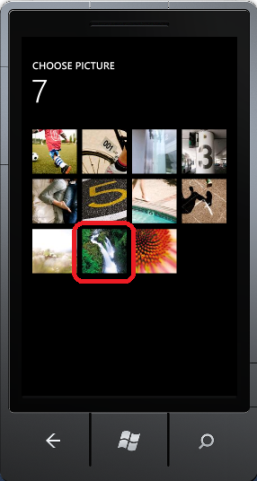
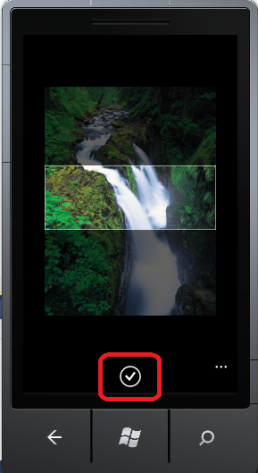
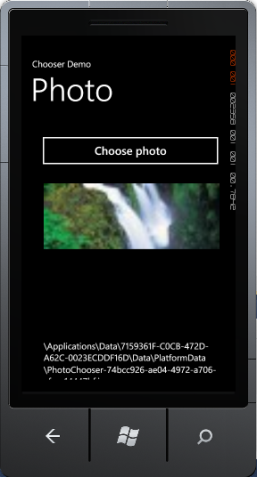
  

图12-18 裁切原始的图像

在初始化时加载PhotoChooserTask的Completed事件处理函数。

Silverlight Project: LaunchersChoosers File: Choosers/ PhotoChooser.xaml.cs

PhotoChooserTask pc;

public PhotoChooser()

{

InitializeComponent();

pc = new PhotoChooserTask();

pc.Completed += new EventHandler<PhotoResult>(pc\_Completed);

}

通过PhotoChooserTask的Show方法调用选择器，设定裁剪相片后的最大高度和宽度，以及将显示Camera的属性设置为True。

Silverlight Project: LaunchersChoosers File: Choosers/ PhotoChooser.xaml.cs

private void button1\_Click(object sender, RoutedEventArgs e)

{

//是否裁剪相片，并设定裁剪相片后的最大高度和宽度

pc.PixelHeight = 30;

pc.PixelWidth = 80;

//设定是否出现拍照的按钮（位于Application Bar）

pc.ShowCamera = true;

pc.Show();

}

PhotoChooserTask的Completed事件处理函数，判断任务返回值，如果为OK则将图片显示在Image控件，将图片名称（包含文件路径）显示在TextBox控件。

Silverlight Project: LaunchersChoosers File: Choosers/ PhotoChooser.xaml.cs

void pc\_Completed(object sender, PhotoResult e)

{

if (e.TaskResult == TaskResult.OK)

{

BitmapImage bmpSource = new BitmapImage();

bmpSource.SetSource(e.ChosenPhoto);

image1.Source = bmpSource;

textBlock1.Text = e.OriginalFileName;

}

else

{

image1.Source = null;

}

}

### 选择器之SaveEmailAddressTask

SaveEmailAddressTask用来储存联络人中 Email 的信息。

初始化SaveEmailAddressTask，并加载Completed事件处理函数；设置Email地址输入框显示手机键盘输入的Email地址。

Silverlight Project: LaunchersChoosers File: Choosers/ SaveEmailAddress.xaml.cs

SaveEmailAddressTask sea;

public SaveEmailAddress()

{

InitializeComponent();

txtEmail.InputScope = new InputScope()

{

Names = { new InputScopeName() { NameValue = InputScopeNameValue.EmailNameOrAddress } }

};

sea = new SaveEmailAddressTask();

sea.Completed += new EventHandler<TaskEventArgs>(sea\_Completed);

}

通过SaveEmailAddressTask的Show方法调用选择器，

Silverlight Project: LaunchersChoosers File: Choosers/ SaveEmailAddress.xaml.cs

private void button1\_Click(object sender, RoutedEventArgs e)

{

sea.Email = txtEmail.Text;

sea.Show();

}

SaveEmailAddressTask的Completed事件处理函数。

Silverlight Project: LaunchersChoosers File: Choosers/ SaveEmailAddress.xaml.cs

void sea\_Completed(object sender, TaskEventArgs e)

{

if (e.TaskResult == TaskResult.OK)

{

//Success

MessageBox.Show("保存成功!");

}

else

{

MessageBox.Show("保存失敗!");

}

}

### 选择器之SavePhoneNumberTask

SavePhoneNumberTask则是用来储存联络人的电话号码。

初始化SavePhoneNumberTask，并加载Completed事件处理函数；设置电话号码输入框显示手机键盘输入的电话号码。

Silverlight Project: LaunchersChoosers File: Choosers/ SavePhoneNumber.xaml.cs

SavePhoneNumberTask spn;

public SavePhoneNumber()

{

InitializeComponent();

txtPhoneNo.InputScope = new InputScope()

{

Names = { new InputScopeName() { NameValue = InputScopeNameValue.TelephoneNumber } }

};

spn = new SavePhoneNumberTask();

spn.Completed += new EventHandler<TaskEventArgs>(spn\_Completed);

}

通过SavePhoneNumberTask的Show方法调用选择器，

Silverlight Project: LaunchersChoosers File: Choosers/ SavePhoneNumber.xaml.cs

private void button1\_Click(object sender, RoutedEventArgs e)

{

spn.PhoneNumber = txtPhoneNo.Text;

spn.Show();

}

SavePhoneNumberTask的Completed事件处理函数。

Silverlight Project: LaunchersChoosers File: Choosers/ SavePhoneNumber.xaml.cs

void spn\_Completed(object sender, TaskEventArgs e)

{

if (e.TaskResult == TaskResult.OK)

{

MessageBox.Show("保存成功!");

}

else

{

MessageBox.Show("保存失敗!");

}

}