Windows Phone Mango编程实践

***Windows Phone Mango Programming Practice***

第二篇 Silverlight交互篇

# 必应地图——导航先锋

子曰：“知者乐水，仁者乐山；知者动，仁者静；知者乐，仁者寿。”

## Windows Phone必应地图概述

必应地图 Silverlight控件结合了Silverlight和Bing Maps的能力，提供了增强的地图体验，支持地理位置服务和地图搜索功能，并支持数据绑定模式。

必应地图控件采用数据绑定在数据模板中处理数据对象的方法，是实现地图控件与后台逻辑低耦合性的最佳做法，尤其是使用MVVM（model-view-ViewModel）架构模式的应用程序。

必应地图控件在使用时需要验证密钥，目前Bing Maps账户中心允许免费创建密钥并使用 Bing Maps控件、Bing Maps SOAP 服务、Bing Maps REST (representational state transfer) 服务以及Bing Spatial Data服务。如果没有有效的密钥，你将不能从Web上获得Bing Maps的内容。

本章的重点内容是地图的图层、图钉和计算路线的实现方法。

## 动手实践——必应地图导航

* 1. 必应地图导航的方式是使用**Bing Maps SOAP**服务来计算从起点至终点的路线，在路线图层来呈现必应地图服务提供的路线行程。
* **Geocode Service** –匹配地址，地点和地理实体到地图上的经度和纬度坐标，并返回一个指定了纬度和经度坐标的位置信息。
* **Route Service** –根据位置和导航点生成路线和驾驶指令，例如包括交通警告和多个地点间的路线提示的指令，以及从所有主要道路通往目的地的指示。
* **Search Service** –分析一个搜索查询，其中包含一个位置或关键字（或两者都有），返回搜索结果。

### 先决条件

申请注册Bing Maps账户。

Bing Maps账户注册地址: [http://www.bingmapsportal.com](http://www.bingmapsportal.com/)。

点击**Create**来使用你的Windows Live ID创建一个新的Bing Maps账户。

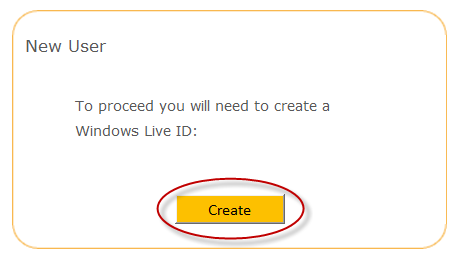


图8-1 Bing Maps注册

填入详细信息，如图8-2 填写Bing maps账户详细信息。

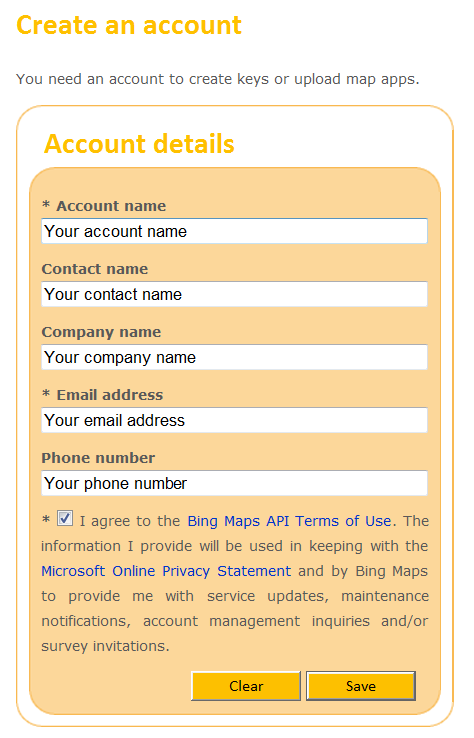


图8-2 填写Bing maps账户详细信息

点击“**Create or view keys**”链接，为应用程序创建一个新的密钥。

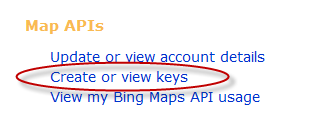


图8-3 创建或显示密钥

填入详细信息，点击**Createkey**创建新密钥。

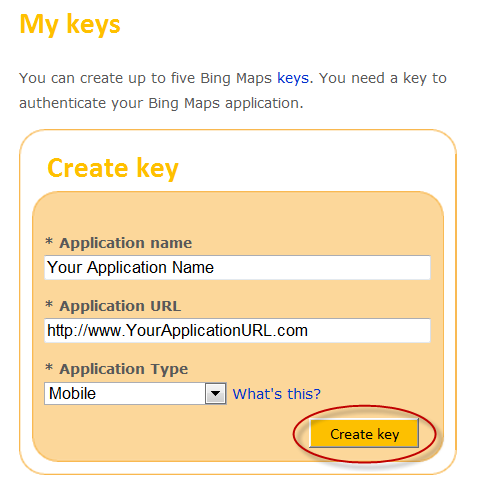


图8-4 创建一个新密钥

### 创建必应地图导航工程

以管理员身份启动**Microsoft Visual Studio 2010**或者**Microsoft Visual Studio 2010 Express**，打开\chapter 08\Begin文件夹下的**Silverlight for Windows Phone**工程“**UsingBingMaps**”。

Windows Phone 应用程序包含Resources 、Helpers文件夹和MainPage页面。图标和样式等资源放在Resources 文件夹中，Helper类放置在Helpers 文件夹中。MainPage页面包含三个文件：

* **MainPage.xaml** –使用XAML 定义Bing maps UI。
* **MainPage.cs** – 这是一个不完整的类，包含执行必应地图启动代码。
* **MainPage.xaml.cs** –包含必应地图控制逻辑。

添加**Microsoft.Phone.Controls.Maps**的引用，如图8-5 添加地图控件的引用。

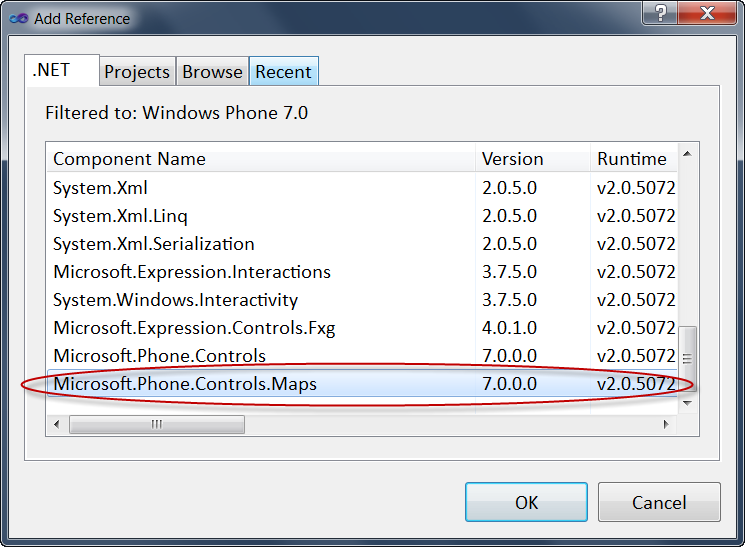


图8-5 添加地图控件的引用

在设计模式下打开**MainPage.xaml**，从工具箱找到Map 控件，拖动到页面中央，并作如下修改。

在**MainPage.xaml**中修改map名称为Map并删除Height和Width属性，这样地图就可以占据整个屏幕；

设置地图模式为航测图模式；

删除Bing 图标和版权，为此将属性**Map.CopyrightVisibility** 和 **Map.LogoVisibility**的值均设置为**Visibility.Collapsed**；

数据绑定和数据模板的使用：绑定**Map.ZoomLevel**，绑定模式为双向绑定；设定**Pushpin**的数据模板和数据绑定。**Pushpin**的数据绑定将在创建图钉（pushpin）图层中讲述。

Silverlight Project: UsingBingMaps File: MainPage.xaml

<!-- Map View -->

<Border x:Name="MapView"

Background="Black"

Height="768" Width="480">

<my:Map Name="Map"

CredentialsProvider="{Binding CredentialsProvider}"

CopyrightVisibility="Collapsed" LogoVisibility="Collapsed"

ZoomLevel="{Binding Zoom, Mode=TwoWay}"

Center="{Binding Center, Mode=TwoWay}" ZoomBarVisibility="Collapsed">

<my:Map.Mode>

<my:AerialMode ShouldDisplayLabels="True" />

</my:Map.Mode>

<my:MapItemsControl ItemsSource="{Binding Routes}">

<my:MapItemsControl.ItemTemplate>

<DataTemplate>

<my:MapPolyline Locations="{Binding Locations}"

Stroke="#FF2C76B7"

Opacity="0.5"

StrokeThickness="6" />

</DataTemplate>

</my:MapItemsControl.ItemTemplate>

</my:MapItemsControl>

<my:MapItemsControl ItemsSource="{Binding Pushpins}">

<my:MapItemsControl.ItemTemplate>

<DataTemplate>

<my:Pushpin Style="{StaticResource PushpinStyle}"

MouseLeftButtonUp="Pushpin\_MouseLeftButtonUp"

Location="{Binding Location}"

Background="{Binding TypeName, Converter={StaticResource PushpinTypeBrushConverter}}">

<Image Source="{Binding Icon}" />

</my:Pushpin>

</DataTemplate>

</my:MapItemsControl.ItemTemplate>

</my:MapItemsControl>

<Button x:Name="ButtonZoomIn"

Style="{StaticResource ButtonZoomInStyle}"

HorizontalAlignment="Left" VerticalAlignment="Top"

Height="56" Width="56" Margin="8,180,0,0"

Click="ButtonZoomIn\_Click" />

<Button x:Name="ButtonZoomOut"

Style="{StaticResource ButtonZoomOutStyle}"

HorizontalAlignment="Left" VerticalAlignment="Top"

Height="56" Width="56" Margin="8,260,0,0"

Click="ButtonZoomOut\_Click" />

</my:Map>

</Border>

### 绑定密钥

打开**App.xaml.cs** 文件并在此类中添加一个内部字符串常量字段 ，命名为”ID”这个字段用于保存你创建的 Bing Maps 私钥。

Silverlight Project: UsingBingMaps File: App.xaml.cs

/// <value>Registered ID used to access map control and Bing maps service.</value>

internal const string Id = "replace-with-your-private-key";

打开 **MainPage.xaml.cs** 文件并在类声明上面添加以下using 语句。

Silverlight Project: UsingBingMaps File: MainPage.xaml.cs

using System;

using System.Device.Location;

using Microsoft.Phone.Controls.Maps;

将密钥绑定到map控件，为此在**MainPage**类中创建一个新的私钥只读字段类型**Microsoft.Phone.Controls.Maps.CredentialsProvider**的实例，并设置读取该实例的方法为Public类型。

Silverlight Project: UsingBingMaps File: MainPage.xaml.cs

/// <value>Provides credentials for the map control.</value>

private readonly CredentialsProvider \_credentialsProvider = new ApplicationIdCredentialsProvider(App.Id);

/// <summary>

/// Gets the credentials provider for the map control.

/// </summary>

public CredentialsProvider CredentialsProvider

{

get { return \_credentialsProvider; }

}

在XAML编辑器中打开 **MainPage.xaml**，将CredentialsProvider方法绑定到**Map.CredentialsProvider**属性。

Silverlight Project: UsingBingMaps File: MainPage.xaml

<my:Map Name="Map"

CredentialsProvider="{Binding CredentialsProvider}"

CopyrightVisibility="Collapsed" LogoVisibility="Collapsed"

ZoomLevel="{Binding Zoom, Mode=TwoWay}"

Center="{Binding Center, Mode=TwoWay}" ZoomBarVisibility="Collapsed">

<my:Map.Mode>

<my:AerialMode ShouldDisplayLabels="True" />

</my:Map.Mode>

**注意**

MainPage.DataContext由MainPage实例本身在MainPage.cs文件中设置，所以隐式的绑定源就是MainPage实例本身。

### 航测图模式和路线图模式

实现航测图模式和路线图模式的切换。

在**MainPage.xaml.cs**中，找到**MainPage** 类中**ChangeMapMode**方法。通过建立**Microsoft.Phone.Controls.Maps.AerialMode**或者**Microsoft.Phone.Controls.Maps.RoadMode**类型的新实例来设置地图模式的属性。

Silverlight Project: UsingBingMaps File: MainPage.xaml.cs

private void ChangeMapMode()

{

if (Map.Mode is AerialMode)

{

Map.Mode = new RoadMode();

}

else

{

Map.Mode = new AerialMode(true);

}

}

### 设定地图中心位置

添加**System.Device**组件的引用，如图8-6 添加**System.Device**组件的引用。

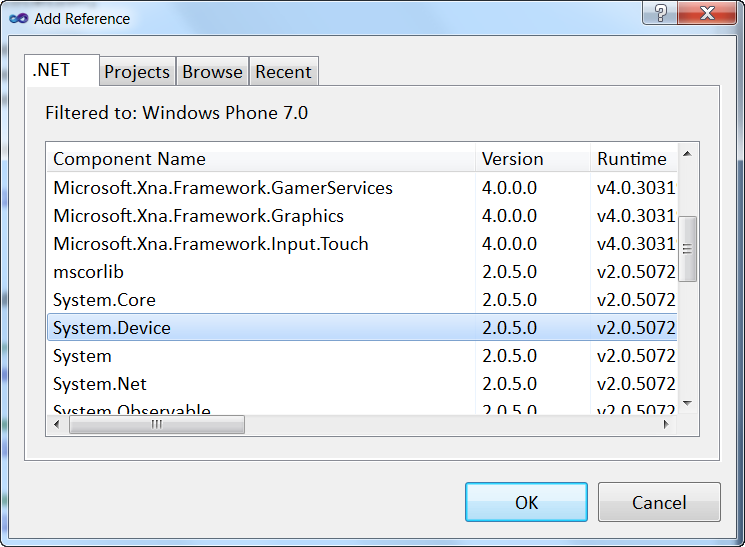


图8-6 添加System.Device 组件的引用

在**MainPage.xaml.cs**中，添加命名为**DefaultLocation** 的**System.Device.Location.GeoCoordinate**类型的默认地理位置变量，并使用设定的经度和纬度坐标来实例化它。

子曰：“知者乐水，仁者乐山；知者动，仁者静；知者乐，仁者寿。”本例将依山傍海、亦动亦静的青岛标志建筑——栈桥的地理位置坐标作为默认的地理位置。

Silverlight Project: UsingBingMaps File: MainPage.xaml.cs

/// <value>Default location coordinate.</value>

//Zhanqiao, Qingdao

private static readonly GeoCoordinate DefaultLocation = new GeoCoordinate(36.05826726951574, 120.31532406806946);

在**MainPage.xaml.cs**中，添加一个新的**System.Device.Location.GeoCoordinate**类型的字段，命名为 \_center。创建名称为**Center**属性来公开**\_center**字段。在其**set**函数中，调用关注属性的变化的方法**NotifyPropertyChanged**。

Silverlight Project: UsingBingMaps File: MainPage.xaml.cs

/// <value>Map center coordinate.</value>

private GeoCoordinate \_center;

/// <summary>

/// Gets or sets the map center location coordinate.

/// </summary>

public GeoCoordinate Center

{

get { return \_center; }

set

{

if (\_center != value)

{

\_center = value;

NotifyPropertyChanged("Center");

}

}

}

在**MainPage.xaml**中，绑定**Map.Center**属性到您刚刚创建的**Center** 属性，绑定模式为双向。

Silverlight Project: UsingBingMaps File: MainPage.xaml

<my:Map Name="Map"

CredentialsProvider="{Binding CredentialsProvider}"

CopyrightVisibility="Collapsed" LogoVisibility="Collapsed"

ZoomLevel="{Binding Zoom, Mode=TwoWay}"

Center="{Binding Center, Mode=TwoWay}" ZoomBarVisibility="Collapsed">

设置应用程序栏的“**Origin**”按钮的处理函数**CenterLocation**方法。在**MainPage.xaml.cs**中找到**CenterLocation** 方法，添加如下代码实现设置Center和Zoom为默认值。

Silverlight Project: UsingBingMaps File: MainPage.xaml.cs

private void CenterLocation()

{

// Center map to default location.

Center = DefaultLocation;

// Reset zoom default level.

Zoom = DefaultZoomLevel;

}

### 创建图钉（pushpin）图层

创建一个名称为**Models**的新文件夹，在**Models**文件夹下创建一个新的公用类**PushpinModel**。此类将呈现图钉数据。

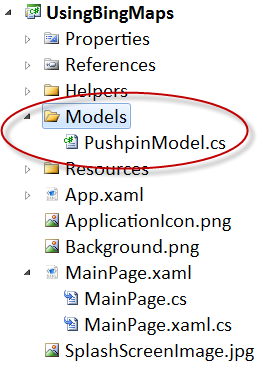
* + 1. 

图8-7 图钉模型

为**PushpinModel**添加一个名称为**Location**新的公共属性，类型为**System.Device.Location.GeoCoordinate PushpinModel**。这个属性表示图钉在地图上的地理位置。

Silverlight Project: UsingBingMaps File: Models/PushpinModel.cs

/// <summary>

/// Gets or sets the pushpin location.

/// </summary>

public GeoCoordinate Location { get; set; }

在**MainPage.xaml.cs**中添加一个名称为**\_pushpins**的私有只读字段，类型为**ObservableCollection<PushpinModel>**，使用默认地理位置的**PushpinModel** 实例对它进行初始化。

Silverlight Project: UsingBingMaps File: MainPage.xaml.cs

/// <value>Collection of pushpins available on map.</value>

private readonly ObservableCollection<PushpinModel> \_pushpins = new ObservableCollection<PushpinModel>

{

new PushpinModel

{

Location = DefaultLocation,

Icon = new Uri("/Resources/Icons/Pushpins/PushpinLocation.png", UriKind.Relative)

}

};

使用**public**属性的方法**Pushpins**公布**\_pushpins**字段，MainPage.xaml使用此方法为图钉的数据模板**MapItemsControl** 绑定**Pushpins**集合，在数据模板中设置**MapItemsControl.ItemTemplate**属性，添加图钉的**DataTemplate** 元素。为在地图上准确的放置图钉，需要绑定**Pushpin.Location** 属性到**PushpinModel**的**Location**属性。

Silverlight Project: UsingBingMaps File: MainPage.xaml.cs

/// <summary>

/// Gets a collection of pushpins.

/// </summary>

public ObservableCollection<PushpinModel> Pushpins

{

get { return \_pushpins; }

}

在MainPage.xaml中，使用图钉样式。设置图钉的内容为**Image** 元素并绑定**Image.Source** 属性到**PushpinModel.Icon** 属性。

Silverlight Project: UsingBingMaps File: MainPage.xaml

<my:MapItemsControl ItemsSource="{Binding Pushpins}">

<my:MapItemsControl.ItemTemplate>

<DataTemplate>

<my:Pushpin Style="{StaticResource PushpinStyle}"

MouseLeftButtonUp="Pushpin\_MouseLeftButtonUp"

Location="{Binding Location}"

Background="{Binding TypeName, Converter={StaticResource PushpinTypeBrushConverter}}">

<Image Source="{Binding Icon}" />

</my:Pushpin>

</DataTemplate>

</my:MapItemsControl.ItemTemplate>

</my:MapItemsControl>

### 定制图钉样式

创建和修改图钉样式和控件模板，打开位于工程文件夹**Resources/Styles** 中的**DefaultStyle.xaml**资源字典，并映射Bing maps控件命名空间。在**DefaultStyle.xaml**文件的上部加入如下的代码。

Silverlight Project: UsingBingMaps File: DefaultStyle.xaml

xmlns:m="clr-namespace:Microsoft.Phone.Controls.Maps;assembly=Microsoft.Phone.Controls.Maps"

使用Microsoft Expression Blend for Windows Phone可以轻松地创建控件模板，该工具是Windows Phone开发工具的一部分。

在**DefaultStyle.xaml**资源字典中为图钉创建一个新的样式如下。

Silverlight Project: UsingBingMaps File: DefaultStyle.xaml

<Style x:Key="ItineraryPushpinStyle" TargetType="m:Pushpin">

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="m:Pushpin">

<Grid Height="20" Width="20">

<VisualStateManager.VisualStateGroups>

<VisualStateGroup x:Name="VisualStateGroup">

<VisualStateGroup.Transitions>

<VisualTransition GeneratedDuration="0:0:0.1">

<VisualTransition.GeneratedEasingFunction>

<PowerEase EasingMode="EaseIn"/>

</VisualTransition.GeneratedEasingFunction>

</VisualTransition>

<VisualTransition GeneratedDuration="0:0:0.1" To="Selected">

<VisualTransition.GeneratedEasingFunction>

<PowerEase EasingMode="EaseIn"/>

</VisualTransition.GeneratedEasingFunction>

</VisualTransition>

</VisualStateGroup.Transitions>

<VisualState x:Name="UnSelected"/>

<VisualState x:Name="Selected">

<Storyboard>

<ColorAnimation Duration="0" To="White" Storyboard.TargetProperty="(Shape.Fill).(SolidColorBrush.Color)" Storyboard.TargetName="ellipse" d:IsOptimized="True"/>

<DoubleAnimation Duration="0" To="1.3" Storyboard.TargetProperty="(UIElement.RenderTransform).(CompositeTransform.ScaleX)" Storyboard.TargetName="ellipse" d:IsOptimized="True"/>

<DoubleAnimation Duration="0" To="1.3" Storyboard.TargetProperty="(UIElement.RenderTransform).(CompositeTransform.ScaleY)" Storyboard.TargetName="ellipse" d:IsOptimized="True"/>

<ColorAnimation Duration="0" To="#FFF08609" Storyboard.TargetProperty="(Shape.Fill).(SolidColorBrush.Color)" Storyboard.TargetName="ellipse\_Center" d:IsOptimized="True"/>

<DoubleAnimation Duration="0" To="1.5" Storyboard.TargetProperty="(UIElement.RenderTransform).(CompositeTransform.ScaleX)" Storyboard.TargetName="ellipse\_Center" d:IsOptimized="True"/>

<DoubleAnimation Duration="0" To="1.5" Storyboard.TargetProperty="(UIElement.RenderTransform).(CompositeTransform.ScaleY)" Storyboard.TargetName="ellipse\_Center" d:IsOptimized="True"/>

</Storyboard>

</VisualState>

</VisualStateGroup>

</VisualStateManager.VisualStateGroups>

<Ellipse x:Name="ellipse" Style="{StaticResource MapPoint}" Width="20" Height="20" RenderTransformOrigin="0.5,0.5" Fill="White" Stroke="#FF2C76B7" StrokeThickness="3" >

<Ellipse.RenderTransform>

<CompositeTransform/>

</Ellipse.RenderTransform>

</Ellipse>

<Ellipse x:Name="ellipse\_Center" Style="{StaticResource MapPoint}" Width="8" Height="8" RenderTransformOrigin="0.5,0.5" Fill="Black" Stroke="{x:Null}" StrokeThickness="2" >

<Ellipse.RenderTransform>

<CompositeTransform/>

</Ellipse.RenderTransform>

</Ellipse>

</Grid>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

**注意**

如果你需要设计自己的图钉控件模板，你应该了解当改变地图显示比例（变焦）时，图钉默认的关联点是在左下角而不是中心或者左上角。

在**PushpinModel**中添加一个**Uri**类型的公用属性，命名为**Icon**，以此来绑定图钉的图像。

Silverlight Project: UsingBingMaps File: Models/PushpinModel.cs

namespace UsingBingMaps.Models

{

/// <summary>

/// Represents a pushpin data model.

/// </summary>

public class PushpinModel

{

/// <summary>

/// Gets or sets the pushpin location.

/// </summary>

public GeoCoordinate Location { get; set; }

/// <summary>

/// Gets or sets the pushpin icon uri.

/// </summary>

public Uri Icon { get; set; }

/// <summary>

/// Gets or sets the pushpin type name.

/// </summary>

public string TypeName { get; set; }

public PushpinModel Clone(GeoCoordinate location)

{

return new PushpinModel

{

Location = location,

TypeName = TypeName,

Icon = Icon

};

}

}

}

在**Resources\Icons** 文件夹中，创建一个名为**Pushpins**的新文件夹，把**\Assets\Resources\Pushpins**文件夹下的所有的图像拷贝到此处，并添加到工程中，如图8-8 图钉图像。

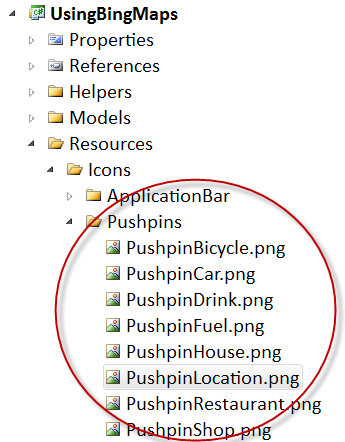


图8-8 图钉图像

将每一个图标的编译动作（**Build Action**）属性修改为**Content**。选择所有的图标，点击右键，选择属性**Properties**，设定**Build Action**属性为**Content**，**Copy to Output Directory**属性为**Do not copy**。如图8-9 修改图标的编译动作。

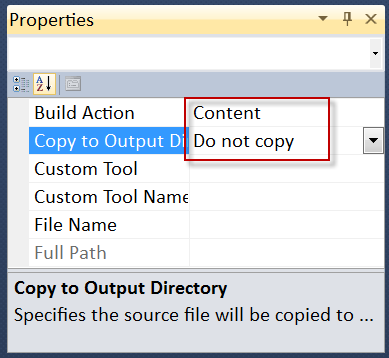


图8-9 修改图标的编译动作

在MainPage.xaml.cs中，设置图钉实例的**PushpinData.Icon**属性，关联**PushpinLocation.png** 图标文件的Uri。

Silverlight Project: UsingBingMaps File: MainPage.xaml.cs

/// <value>Collection of pushpins available on map.</value>

private readonly ObservableCollection<PushpinModel> \_pushpins = new ObservableCollection<PushpinModel>

{

new PushpinModel

{

Location = DefaultLocation,

Icon = new Uri("/Resources/Icons/Pushpins/PushpinLocation.png", UriKind.Relative)

}

};

### 变焦按钮

添加自定义的变焦按钮，实现放大或缩小地图。

在MainPage.xaml.cs文件中，添加一个新的private字段，类型为double。

Silverlight Project: UsingBingMaps File: MainPage.xaml.cs

/// <value>Map zoom level.</value>

private double \_zoom;

设置限制变焦倍数的double常量。

Silverlight Project: UsingBingMaps File: MainPage.xaml.cs

/// <value>Default map zoom level.</value>

private const double DefaultZoomLevel = 16.0;

/// <value>Maximum map zoom level allowed.</value>

private const double MaxZoomLevel = 21.0;

/// <value>Minimum map zoom level allowed.</value>

private const double MinZoomLevel = 1.0;

在MainPage.xaml.cs中设定double类型**Zoom**的get和set属性。在set函数中，要考虑变焦倍数的上限和下限的限制并关注属性的变化。

Silverlight Project: UsingBingMaps File: MainPage.xaml.cs

/// <summary>

/// Gets or sets the map zoom level.

/// </summary>

public double Zoom

{

get { return \_zoom; }

set

{

var coercedZoom = Math.Max(MinZoomLevel, Math.Min(MaxZoomLevel, value));

if (\_zoom != coercedZoom)

{

\_zoom = value;

NotifyPropertyChanged("Zoom");

}

}

}

在XAML编辑器中打开MainPage.xaml并绑定**Zoom**属性到**Map.ZoomLevel**，绑定模式为TwoWay双向绑定。

Silverlight Project: UsingBingMaps File: MainPage.xaml

<my:Map Name="Map"

CredentialsProvider="{Binding CredentialsProvider}"

CopyrightVisibility="Collapsed" LogoVisibility="Collapsed"

ZoomLevel="{Binding Zoom, Mode=TwoWay}"

Center="{Binding Center, Mode=TwoWay}" ZoomBarVisibility="Collapsed">

从工具箱中拖动两个按钮到地图控件的中间左侧，一个用于放大，另一个用于缩小。注册相应按钮的Click事件到ButtonZoomIn\_Click和ButtonZoomOut\_Click事件处理函数。设置放大的变焦按钮的样式**Button.Style** 属性为**ButtonZoomInStyle** ，缩小的变焦按钮的样式**Button.Style** 属性为**ButtonZoomOutStyle** 样式。**ButtonZoomInStyle**和**ButtonZoomOutStyle** 样式定义在DefaultStyle.xaml 资源文件中。

Silverlight Project: UsingBingMaps File: MainPage.xaml

<Button x:Name="ButtonZoomIn"

Style="{StaticResource ButtonZoomInStyle}"

HorizontalAlignment="Left" VerticalAlignment="Top"

Height="56" Width="56" Margin="8,180,0,0"

Click="ButtonZoomIn\_Click" />

<Button x:Name="ButtonZoomOut"

Style="{StaticResource ButtonZoomOutStyle}"

HorizontalAlignment="Left" VerticalAlignment="Top"

Height="56" Width="56" Margin="8,260,0,0"

Click="ButtonZoomOut\_Click" />

**ButtonZoomIn\_Click**和**ButtonZoomOut\_Click**事件处理函数已经包含在MainView.cs 文件中， **ButtonZoomIn\_Click**事件处理函数调用时将**Zoom**属性的值加1，**ButtonZoomOut\_Click** 事件处理函数调用时则将**Zoom**属性的值减1。

Silverlight Project: UsingBingMaps File: MainPage.cs

private void ButtonZoomIn\_Click(object sender, System.Windows.RoutedEventArgs e)

{

Zoom += 1;

}

private void ButtonZoomOut\_Click(object sender, System.Windows.RoutedEventArgs e)

{

Zoom -= 1;

}

### 横向和纵向视图的设定

为实现Windows Phone横向和纵向的切换，需要做以下代码修改。设置MainPage.xaml支持的显示属性为即支持横向显示也支持纵向显示，初始显示模式为纵向。当Windows Phone感应器自动识别手机的方向，当方向发生变化时会自动切换，不需要开发者干预。

修改MainPage.xaml文件的SupportedOrientations属性。

Silverlight Project: UsingBingMaps File: MainPage.xaml

SupportedOrientations="PortraitOrLandscape" Orientation="Portrait"

添加MainPage.xaml文件的OrientationChanged事件处理函数。打开ManPage.xaml的Properties属性选项卡，在Events选项页增加OrientationChanged事件处理函数PhoneApplicationPage\_OrientationChanged。如图8-10 添加事件处理函数。

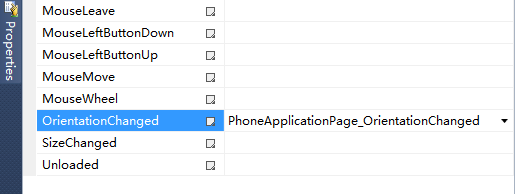


图8-10 事件处理函数

在MainPage.xaml文件选中OrientationChanged事件处理函数PhoneApplicationPage\_OrientationChanged，点击右键，选择“Navigate to Event Handler”，Visual Studio将导航到MainPage.xaml.cs文件的PhoneApplicationPage\_OrientationChanged代码。如图8-11 导航所示，Visual Studio将导航MainPage.xaml.cs。

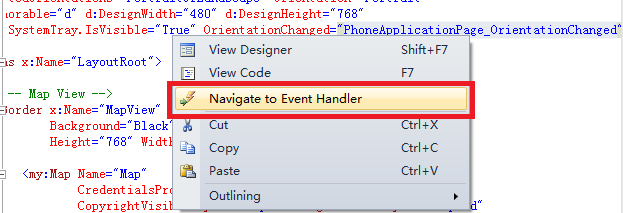


图8-11 导航

设置横向视图和纵向视图时XAML的高度和宽度，以及变焦按钮的位置，以免变焦按钮与应用程序栏重叠。

Silverlight Project: UsingBingMaps File: MainPage.xaml.cs

private void PhoneApplicationPage\_OrientationChanged(object sender, OrientationChangedEventArgs e)

{

// Switch Map View Border's Height and Width based on an orientation change.

if ((e.Orientation & PageOrientation.Portrait) == (PageOrientation.Portrait))

{

MapView.Height = 768;

MapView.Width = 480;

ButtonZoomIn.Margin = new Thickness(8, 180, 0, 0);

ButtonZoomOut.Margin = new Thickness(8, 260, 0, 0);

}

else

{

MapView.Height = 480;

MapView.Width = 768;

ButtonZoomIn.Margin = new Thickness(320, 0, 0, 0);

ButtonZoomOut.Margin = new Thickness(400, 0, 0, 0);

}

}

### 计算导航路线

添加**Bing Maps Geocode**服务的引用，为此右键点击工程**UsingBingMaps**的**References**，然后选择**Add Service Reference…**。

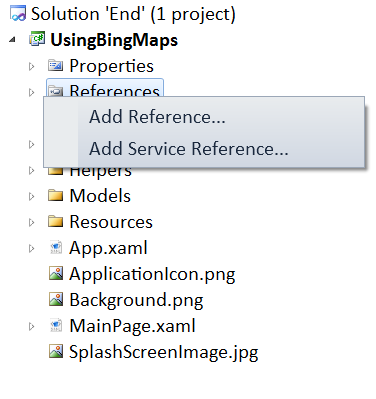
* + 1. 

图8-11 添加服务的引用

把下面的服务地址粘贴到打开的对话框，然后点击Go: <http://dev.virtualearth.net/webservices/v1/geocodeservice/geocodeservice.svc>

在Namespace字段，输入Bing.Geocode，然后点击OK，如图8-12 必应地图Geocode服务。

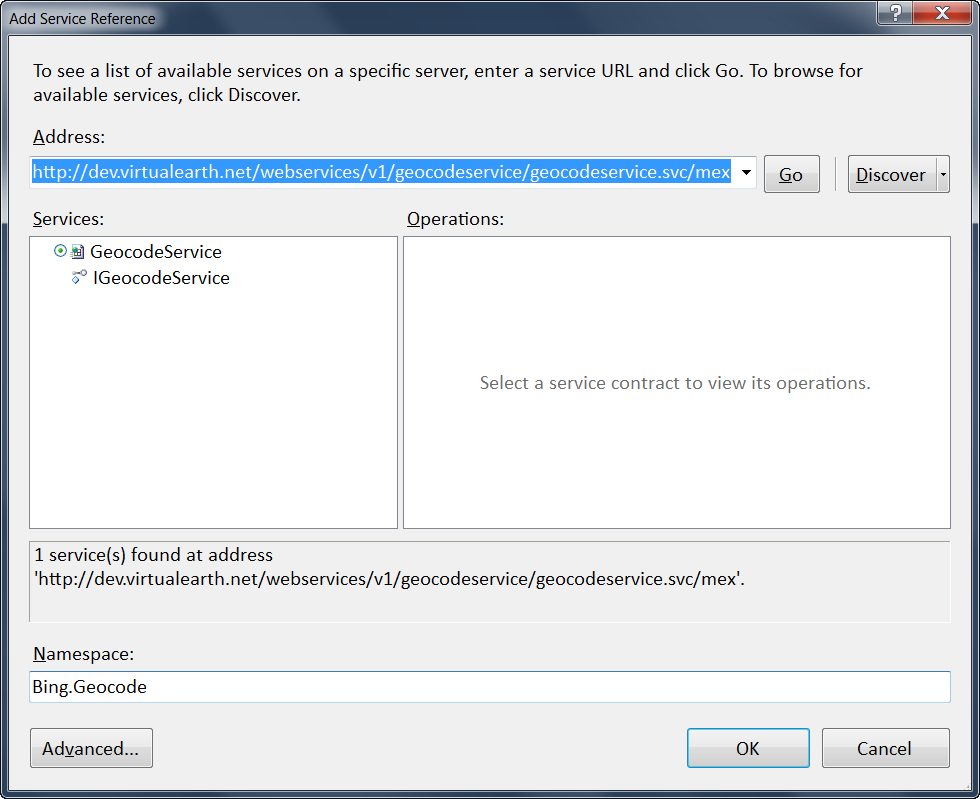


图8-12 必应地图**Geocode**服务

添加**Bing Maps Route**服务的引用，为此右键点击工程**UsingBingMaps**的**References**，然后选择**Add Service Reference…**。

把下面的服务地址粘贴到打开的对话框，然后点击Go: <http://dev.virtualearth.net/webservices/v1/routeservice/routeservice.svc>

在Namespace字段，输入**Bing. Route**，然后点击OK。

添加到**Bing Maps Search**服务的引用，为此右键点击工程**UsingBingMaps**的**References**，然后选择**Add Service Reference…**。

把下面的服务地址粘贴到打开的对话框，然后点击Go: <http://dev.virtualearth.net/webservices/v1/searchservice/searchservice.svc>

在**Namespace**字段，输入**Bing. Search**， 然后点击OK。

打开 **ServiceReferences.ClientConfig** 配置文件，作为项目的一项配置它一般会自动创建 ，删除文件中所有与**CustomBinding** 相关的内容，配置文件如下。

Silverlight Project: UsingBingMaps File: ServiceReferences.ClientConfig

<configuration>

<system.serviceModel>

<bindings>

<basicHttpBinding>

<binding name="BasicHttpBinding\_IGeocodeService" maxBufferSize="2147483647"

maxReceivedMessageSize="2147483647">

<security mode="None" />

</binding>

<binding name="BasicHttpBinding\_IRouteService" maxBufferSize="2147483647"

maxReceivedMessageSize="2147483647">

<security mode="None" />

</binding>

<binding name="BasicHttpBinding\_ISearchService" maxBufferSize="2147483647"

maxReceivedMessageSize="2147483647">

<security mode="None" />

</binding>

</basicHttpBinding>

</bindings>

<client>

<endpoint address="http://dev.virtualearth.net/webservices/v1/geocodeservice/GeocodeService.svc"

binding="basicHttpBinding" bindingConfiguration="BasicHttpBinding\_IGeocodeService"

contract="Bing.Geocode.IGeocodeService" name="BasicHttpBinding\_IGeocodeService" />

<endpoint address="http://dev.virtualearth.net/webservices/v1/routeservice/routeservice.svc"

binding="basicHttpBinding" bindingConfiguration="BasicHttpBinding\_IRouteService"

contract="Bing.Route.IRouteService" name="BasicHttpBinding\_IRouteService" />

<endpoint address="http://dev.virtualearth.net/webservices/v1/searchservice/searchservice.svc"

binding="basicHttpBinding" bindingConfiguration="BasicHttpBinding\_ISearchService"

contract="Bing.Search.ISearchService" name="BasicHttpBinding\_ISearchService" />

</client>

</system.serviceModel>

</configuration>

创建辅助类，通过异步调用服务来计算路线。将**RouteCalculator.cs**、**RoutingState.cs**和**RouteCalculationError.cs**文件添加到**Helpers** 工程文件夹，如图 路线计算类。可以在**Assets/Code**目录中找到这些文件。

**RouteCalculator** 类公开了一个**CalculateAsync**公用方法和一个公用的**Error**事件。初始化一个带有以下参数RouteCalculator类型的新实例，如果发生错误，错误事件会由UI线程抛出。

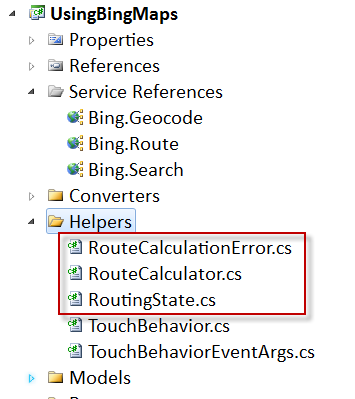


图8-13 路线计算类

向**Models**工程文件夹中添加一个模板公用类**RouteModel**来表现路线数据。在**RouteModel** 类中添加一个新的**collection**来保存路线坐标系，类型为**Microsoft.Phone.Controls.Maps.LocationCollection**，作为一个公用属性公开并命名为**Locations**。

创建一个构造函数，它有一个类型为**Microsoft.Phone.Controls.Maps.Platform.Location**的参数。使用它来初始化内部集合，**Location**有一个到**GeoCoordinate**类型的隐式转换。

Silverlight Project: UsingBingMaps File: Models/RouteModel.cs

public class RouteModel

{

private readonly LocationCollection \_locations;

/// <summary>

/// Gets the location collection of this route.

/// </summary>

public ICollection<GeoCoordinate> Locations

{

get { return \_locations; }

}

/// <summary>

/// Initializes a new instance of this type.

/// </summary>

/// <param name="locations">A collection of locations.</param>

public RouteModel(ICollection<Location> locations)

{

\_locations = new LocationCollection();

foreach (Location location in locations)

{

\_locations.Add(location);

}

}

}

在**MainPage.xaml.cs**中，添加一个新的**ObservableCollection<RouteModel>** 并作为公用属性公开，命名为**Routes**。这个集合保存计算后的路线并绑定到路线图层。

Silverlight Project: UsingBingMaps File: MainPage.xaml.cs

/// <value>Collection of calculated map routes.</value>

private readonly ObservableCollection<RouteModel> \_routes = new ObservableCollection<RouteModel>();

/// <summary>

/// Gets a collection of routes.

/// </summary>

public ObservableCollection<RouteModel> Routes

{

get { return \_routes; }

}

添加两个公用 string 属性：**To**和**From**，即起点和终点。

Silverlight Project: UsingBingMaps File: MainPage.xaml.cs

/// <summary>

/// Gets or sets the route destination location.

/// </summary>

public string To { get; set; }

/// <summary>

/// Gets or sets the route origin location.

/// </summary>

public string From { get; set; }

**CalculateRoute** 方法并使用路线计算器辅助类计算一个路线：创建**try/catch**块，在捕捉到异常时使用 **MessageBox**显示错误消息。

为实现异步计算路线，在try块中创建一个**RouteCalculator**实例。当路线计算完毕时，清除Route集合。并注册错误消息给**RouteCalculator.Error**事件，最后调用**RouteCalculator.CalculateAsync**方法来开始计算路线。

以路线计算器参数为基础创建一个新的**RouteModel**实例，并向路线集合中添加新的路线。以新路线为中心显示地图， 通过调用**Map.SetView**传递**LocationRect** 实例。**LocationRect** 实例使用LocationRect.CreateLocationRect方法创建。

Silverlight Project: UsingBingMaps File: MainPage.xaml.cs

private void CalculateRoute()

{

try

{

var routeCalculator = new RouteCalculator(

CredentialsProvider,

To,

From,

Dispatcher,

result =>

{

// Clear the route collection to have only one route at a time.

Routes.Clear();

// Create a new route based on route calculator result,

// and add the new route to the route collection.

var routeModel = new RouteModel(result.Result.RoutePath.Points);

Routes.Add(routeModel);

// Set the map to center on the new route.

var viewRect = LocationRect.CreateLocationRect(routeModel.Locations);

Map.SetView(viewRect);

});

// Display an error message in case of fault.

routeCalculator.Error += r => MessageBox.Show(r.Reason);

// Start the route calculation asynchronously.

routeCalculator.CalculateAsync();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

在**MainPage.xaml**中添加起点和终点的文本框。找到**RouteView** border并添加一个新**Grid**子项，由两个**TextBlock**s的组成：From和To。From和To控件使用双向绑定模式绑定。在Grid中添加一个“Go” 按钮，设置它的样式为 **ButtonGoStyle** ，添加点击事件的处理函数**ButtonGo\_Click**。

Silverlight Project: UsingBingMaps File: MainPage.xaml

<!-- Route View -->

<Border x:Name="RouteView"

Height="160" Margin="0"

Padding="8" RenderTransformOrigin="0.5,0.5" Width="480"

Background="{StaticResource ControlBackgroundBrush}">

<Border.RenderTransform>

<CompositeTransform TranslateY="-160"/>

</Border.RenderTransform>

<Grid>

<Grid.RowDefinitions>

<RowDefinition />

<RowDefinition />

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="50" />

<ColumnDefinition Width="0.8\*" />

<ColumnDefinition Width="0.2\*"/>

</Grid.ColumnDefinitions>

<TextBlock Text="From" Grid.Row="0" Grid.Column="0" VerticalAlignment="Center" />

<TextBox Text="{Binding From, Mode=TwoWay}" Grid.Row="0" Grid.Column="1" Grid.ColumnSpan="2" />

<TextBlock Text="To" Grid.Row="1" Grid.Column="0" VerticalAlignment="Center" />

<TextBox Text="{Binding To, Mode=TwoWay}" Grid.Row="1" Grid.Column="1" />

<Button Content="Go" Grid.Column="2" Grid.Row="1" Click="ButtonGo\_Click" Style="{StaticResource ButtonGoStyle}" />

</Grid>

</Border>

创建路线图层，创建**MapItemsControl**绑定**Routes** 属性。要在地图上用单线绘制路线，用一个数据模板设置**MapItemsControl.ItemTemplate**，该模板包含了**MapPolyline**的实例。用**RouteModel.Locations**属性来绑定 **MapPolyline** 实例。

Silverlight Project: UsingBingMaps File: MainPage.xaml

<my:MapItemsControl ItemsSource="{Binding Routes}">

<my:MapItemsControl.ItemTemplate>

<DataTemplate>

<my:MapPolyline Locations="{Binding Locations}"

Stroke="#FF2C76B7"

Opacity="0.5"

StrokeThickness="6" />

</DataTemplate>

</my:MapItemsControl.ItemTemplate>

</my:MapItemsControl>

### 测试应用程序

1. 变焦控制

在默认状态下切换手机为纵向显示，本例中的默认地点为青岛市的标志建筑—栈桥，点击放大按钮改变焦距俯视伸向大海怀抱的建筑。观看航测图的地图，就像身处地球之外观察我们生存的星球。

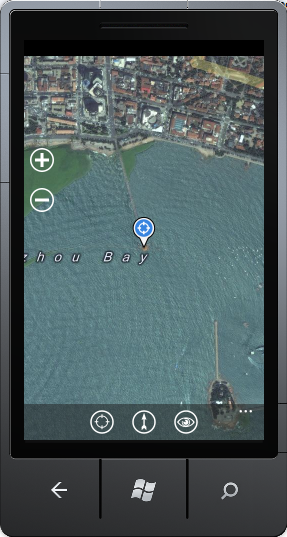
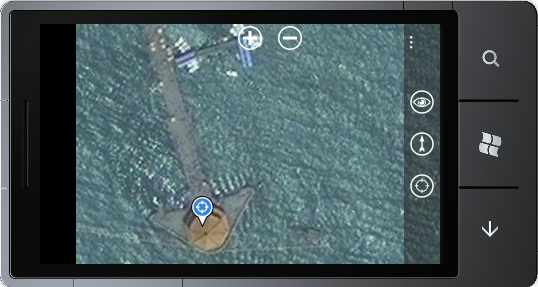
 

图8-14 变焦控制

旋转手机的方向，拖动画面查看地图显示的内容。下图的位置是青岛五四广场的标志——五月的风。

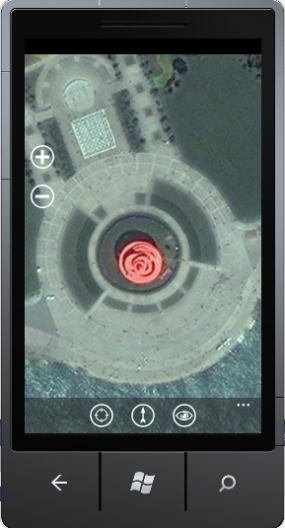
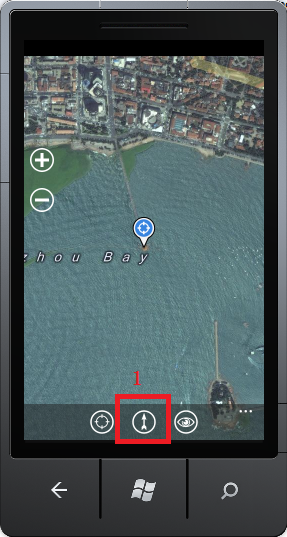
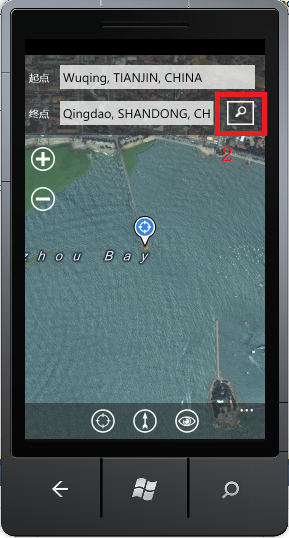
 

图8-15 横向和纵向视图显示

1. 计算路线

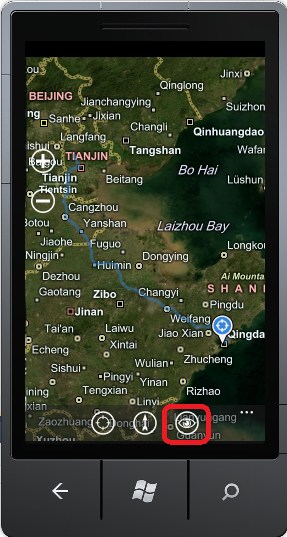
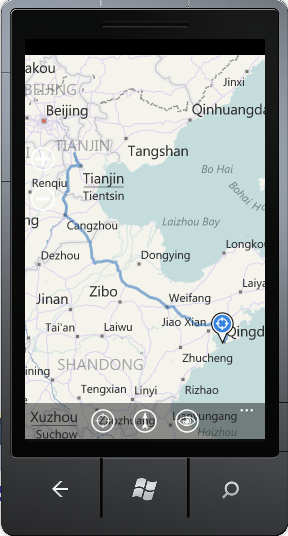
在起点和终点的文本框中输入有效的地址。 默认起点：Wuqing, TIANJIN, CHINA。默认终点：Qingdao, SHANDONG, CHINA。点击搜索图标，程序默认显示天津武清至山东青岛的路线图。如果您输入的地点在地图上找不到，会显示“未找到相关地点的路线”的信息提示。

* 1. 图8-16 计算路线

1. 航测图和路线图模式切换

在应用程序栏点击Mode按钮切换地图显示模式，即航测图模式和路线图模式切换。如图8-17 路线图模式切换至航测图模式。左侧为航测图，右侧为路线图。

* 1.  
  2. 图8-17 路线图模式切换至航测图模式

## 必应地图开发资源

### 开发者资源

任何使用必应地图SOAP服务的疑问，可使用以下帮助资源：

* 必应地图开发者论坛 [Bing Maps Forum](http://social.msdn.microsoft.com/forums/en-US/vemapcontroldev/threads/)
* 阅读必应地图开发者博客 [Bing Maps Developer blog](http://www.bing.com/community/blogs/maps/default.aspx)

### 账户访问问题

如果在使用必应地图开发者账户时遇到问题，请联系必应地图账户管理员[Bing Maps Account Administrator](mailto:mpnet@microsoft.com?subject=Bing%20Maps%20Account%20Access)。

**注意**

请注意必应地图账户访问问题的支持是可用的只是通过电子邮件，电子邮件是在两个工作日内答复。