# Cortana 扩展语音命令的使用

# 动手实验手册

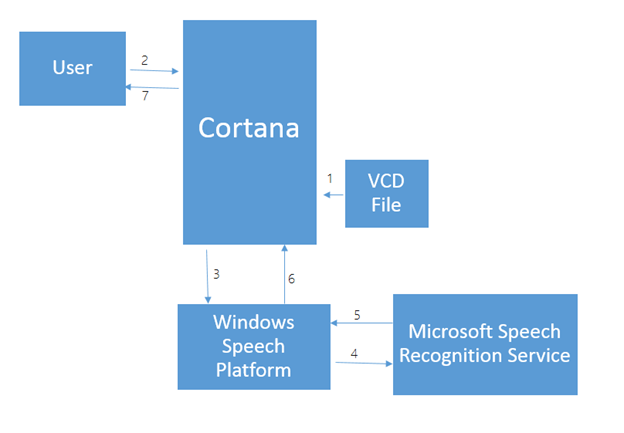
## 概述

Cortana 是一款个人智能助理，它能够了解用户的喜好和习惯，帮助用户进行日程安排、问题解答等。在Windows Phone 8.1发布的时候，Cortana 正式问世，直到Win10 SDK,在语音命令功能上有了大幅度的提升。其中，最大的进步就是从新定义了命令配置文件，支持更为自然的人工语言形式。并且，支持Win10 SDK的新特性，App Service。通过调用App Service就可以在APP 没有前台运行的时候为Cortana提供数据交互。这样一来Cortana语音命令不仅支持与前台APP的交互，还可以同APP的后台服务进行交互。这次升级，Win10 SDK 为开发者提供更加智能的语音命令API，使开发者可以更为轻松的集成语音命令。

## 通过示例您将学会

* Voice Command Definition（VCD）文件元素、属性定义。
* 在Cortana中使用语音命令启动前台应用。
* 在Cortana中使用语音命令启动后台应用。
* 如何在Cortana中与后台应用交互。
* 如何动态地修改语音命令定义（VCD）短语列表。

## 前台语音命令架构



**Cortana的工作流程如下：**

1. 当APP运行时，需要在Cortana中安装VCD文件。（VCD：语音命令定义。是一个XML文件，其中包含了所有的命令，用来激活应用程序）。
2. 用户可以使用在VCD文件中定义的命令，来启动应用程序。
3. （3、4、5、6）用户通过微软云端语音平台和微软语音识别服务将语音转换成命令文本。
4. Cortana接收通过识别的字符串，来确定启动应用程序，并在应用程序中执行相应的事件。

## 挑战

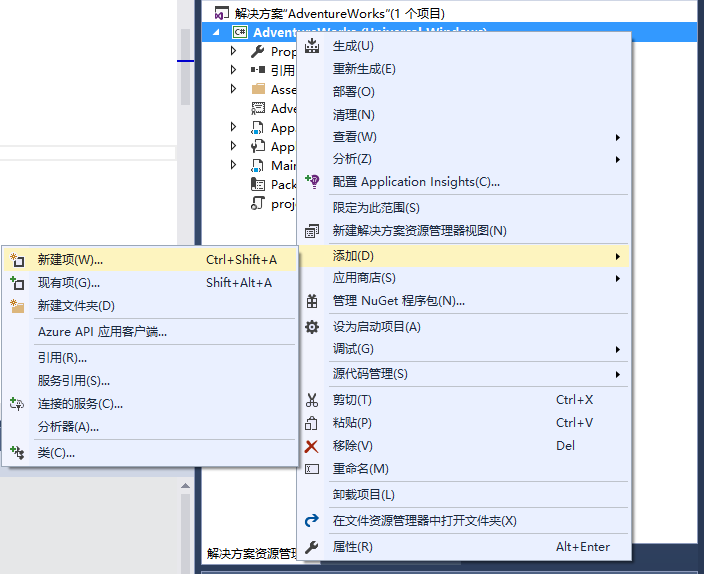
我们将使用Visual Studio 2015 打开AdventureWorks项目，来学习Windows Cortana相关API。

### 场景描述

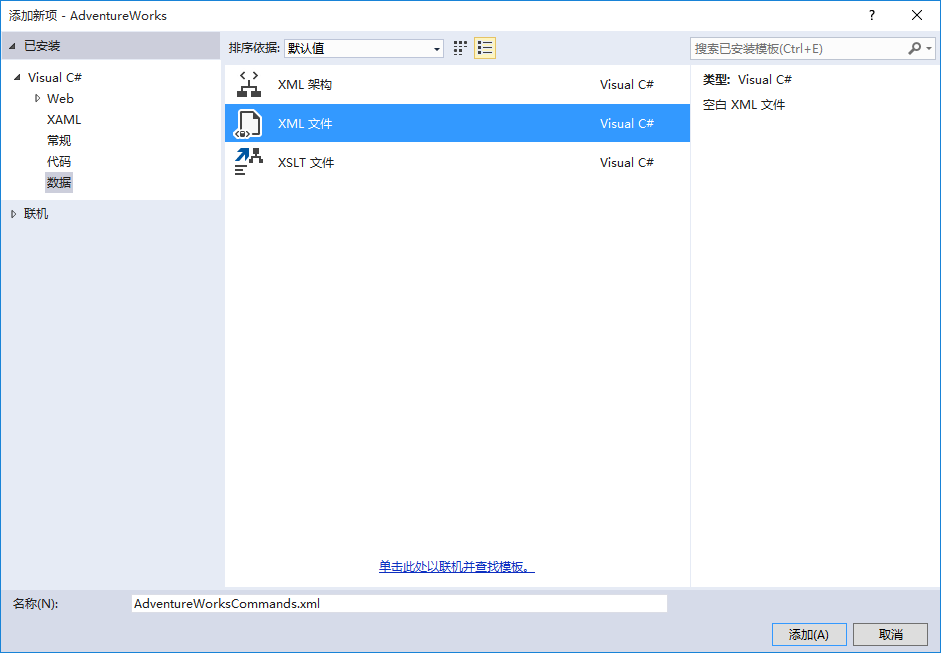
在这个项目中，我们实现了一个关于旅行计划的管理。通过APP用户可以记录旅行的相关信息，如：什么时间段内，在哪个城市，做了什么等。用户通过Cortana的语音命令可以启动前台应用程序，还可以通过语音命令与应用程序的后台服务进行交互。在应用程序中，可以根据语音命令来显示游记的详细信息。为了提高程序的智能程度，应用程序通过当前设备设置的语言来更新命令短语列表。

### 创建Voice Command Definition（VCD）文件

在“解决方案资源管理器”中，右键单击该项目AdventureWorks（除非你有其他的命名）。从“添加”菜单中选择“新建项”



在左侧面板中，单击已安装-数据，选择“XML 文件”，对于这个项目的实验目的，我们建议创建XML 文件的名称为AdventureWorksCommands.xml。点击“添加”按钮。



在“解决方案资源管理器”中，点击AdventureWorksCommands.xml。在“属性”窗口中，设置“复制到输出目录”选项为始终复制，这将确保XML文件是应用程序包的一部分。

编辑AdventureWorksCommands.xml文件，为应用程序添加语音命令，基于我们实验的目的，我们的语音命令定义，如下:

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <VoiceCommands xmlns="http://schemas.microsoft.com/voicecommands/1.2">  <CommandSet xml:lang="en-us" Name="AdventureWorksCommandSet\_en-us">  <CommandPrefix>Adventure Works,</CommandPrefix>  <Example> Show trip to London </Example>  <Command Name="showTripToDestination">  <Example> Show trip to London </Example>  <ListenFor RequireAppName="BeforeOrAfterPhrase"> show [my] trip to {destination} </ListenFor>  <ListenFor RequireAppName="ExplicitlySpecified"> show [my] {builtin:AppName} trip to {destination} </ListenFor>  <Feedback> Showing trip to {destination} </Feedback>  <Navigate />  </Command>  <Command Name="whenIsTripToDestination">  <Example> When is my trip to Las Vegas?</Example>  <ListenFor RequireAppName="BeforeOrAfterPhrase"> when is [my] trip to {destination}</ListenFor>  <ListenFor RequireAppName="ExplicitlySpecified"> when is [my] {builtin:AppName} trip to {destination} </ListenFor>  <Feedback> Looking for trip to {destination}</Feedback>  <VoiceCommandService Target="AdventureWorksVoiceCommandService"/>  </Command>    <Command Name="cancelTripToDestination">  <Example> Cancel my trip to Las Vegas </Example>  <ListenFor RequireAppName="BeforeOrAfterPhrase"> cancel [my] trip to {destination}</ListenFor>  <ListenFor RequireAppName="ExplicitlySpecified"> cancel [my] {builtin:AppName} trip to {destination} </ListenFor>  <Feedback> Cancelling trip to {destination}</Feedback>  <VoiceCommandService Target="AdventureWorksVoiceCommandService"/>  </Command>  <PhraseList Label="destination">  <Item>London</Item>  <Item>Las Vegas</Item>  <Item>Melbourne</Item>  <Item>Yosemite National Park</Item>  </PhraseList>  </CommandSet>  <CommandSet xml:lang="en-gb" Name="AdventureWorksCommandSet\_en-gb">  <CommandPrefix> Adventure Works, </CommandPrefix>  <Example> Show trip to London </Example>  <Command Name="showTripToDestination">  <Example> Show trip to London </Example>  <ListenFor RequireAppName="BeforeOrAfterPhrase"> show [my] trip to {destination} </ListenFor>  <ListenFor RequireAppName="ExplicitlySpecified"> show [my] {builtin:AppName} trip to {destination} </ListenFor>  <Feedback> Showing trip to {destination} </Feedback>  <Navigate Target="foo"/>  </Command>  <Command Name="whenIsTripToDestination">  <Example> When is my trip to Las Vegas?</Example>  <ListenFor RequireAppName="BeforeOrAfterPhrase"> when is [my] trip to {destination}</ListenFor>  <ListenFor RequireAppName="ExplicitlySpecified"> when is [my] {builtin:AppName} trip to {destination} </ListenFor>  <Feedback> Looking for trip to {destination}</Feedback>  <VoiceCommandService Target="AdventureWorksVoiceCommandService"/>  </Command>  <Command Name="cancelTripToDestination">  <Example> Cancel my trip to Las Vegas </Example>  <ListenFor RequireAppName="BeforeOrAfterPhrase"> cancel [my] trip to {destination}</ListenFor>  <ListenFor RequireAppName="ExplicitlySpecified"> cancel [my] {builtin:AppName} trip to {destination} </ListenFor>  <Feedback> Cancelling trip to {destination}</Feedback>  <VoiceCommandService Target="AdventureWorksVoiceCommandService"/>  </Command>  <PhraseList Label="destination">  <Item>London</Item>  <Item>Las Vegas</Item>  <Item>Melbourne</Item>  <Item>Yosemite National Park</Item>  </PhraseList>  </CommandSet>  <CommandSet xml:lang="fr-fr" Name="AdventureWorksCommandSet\_fr-fr">  <CommandPrefix> Adventure Works, </CommandPrefix>  <Example>Montrer mon voyage à Londres </Example>  <Command Name="showTripToDestination">  <Example>Montrer mon voyage à Londres</Example>  <ListenFor RequireAppName="BeforeOrAfterPhrase"> montrer [mon] voyage à {destination}</ListenFor>  <ListenFor RequireAppName="ExplicitlySpecified"> montrer [mon] {builtin:AppName} voyage à {destination} </ListenFor>  <Feedback>Voici les détails du votre voyage à {destination}</Feedback>  <Navigate Target="foo"/>  </Command>  <Command Name="whenIsTripToDestination">  <Example>Quand a lieu mon voyage à Las Vegas?</Example>  <ListenFor RequireAppName="BeforeOrAfterPhrase">quand a lieu [mon] voyage à {destination}</ListenFor>  <ListenFor RequireAppName="ExplicitlySpecified">quand a lieu [mon] {builtin:AppName} voyage à {destination}</ListenFor>  <Feedback>Le chercher pour votre voyage à {destination} est en cours</Feedback>  <VoiceCommandService Target="AdventureWorksVoiceCommandService"/>  </Command>  <Command Name="cancelTripToDestination">  <Example>Annuler mon voyage à Las Vegas</Example>  <ListenFor RequireAppName="BeforeOrAfterPhrase"> annuler [mon] voyage à {destination}</ListenFor>  <ListenFor RequireAppName="ExplicitlySpecified"> annuler [mon] {builtin:AppName} voyage à {destination} </ListenFor>  <Feedback>L'annulation du votre voyage à {destination} est en cours</Feedback>  <VoiceCommandService Target="AdventureWorksVoiceCommandService"/>  </Command>  <PhraseList Label="destination">  <Item>London</Item>  <Item>Melbourne</Item>  <Item>Yosemite National Park</Item>  </PhraseList>  </CommandSet>  <CommandSet xml:lang="it-it" Name="AdventureWorksCommandSet\_it-it">  <CommandPrefix> Adventure Works, </CommandPrefix>  <Example> Mostra il viaggio a Londra </Example>  <Command Name="showTripToDestination">  <Example> Mostra viaggio Londra </Example>  <ListenFor RequireAppName="BeforeOrAfterPhrase"> mostra il [mio] viaggio a {destination}</ListenFor>  <ListenFor RequireAppName="ExplicitlySpecified"> mostra il [mio] viaggio di {builtin:AppName} a {destination}</ListenFor>  <Feedback> Mostro il viaggio a {destination}</Feedback>  <Navigate Target="foo"/>  </Command>  <Command Name="whenIsTripToDestination">  <Example>Quando è il mio viaggio a Las Vegas?</Example>  <ListenFor RequireAppName="BeforeOrAfterPhrase"> quando è il [mio] viaggio a {destination}</ListenFor>  <ListenFor RequireAppName="ExplicitlySpecified"> quando è il [mio] viaggio di {builtin:AppName} a {destination}</ListenFor>  <Feedback> Ricerca del viaggio a {destination} in corso </Feedback>  <VoiceCommandService Target="AdventureWorksVoiceCommandService"/>  </Command>  <Command Name="cancelTripToDestination">  <Example> Annulla il mio viaggio a Las Vegas </Example>  <ListenFor RequireAppName="BeforeOrAfterPhrase"> annulla il [mio] viaggio a {destination} </ListenFor>  <ListenFor RequireAppName="ExplicitlySpecified"> annulla il [mio] viaggio di {builtin:AppName} a {destination}</ListenFor>  <Feedback> Sto annullando il viaggio a {destination} </Feedback>  <VoiceCommandService Target="AdventureWorksVoiceCommandService"/>  </Command>  <PhraseList Label="destination">  <Item>London</Item>  <Item>Melbourne</Item>  <Item>Yosemite National Park</Item>  </PhraseList>  </CommandSet>  <CommandSet xml:lang="de-de" Name="AdventureWorksCommandSet\_de-de">  <CommandPrefix> Adventure Works, </CommandPrefix>  <Example>Zeige Reise nach London</Example>  <Command Name="showTripToDestination">  <Example> Meine Reise nach London zeigen</Example>  <ListenFor RequireAppName="BeforeOrAfterPhrase">[Meine] Reise nach {destination} zeigen.</ListenFor>  <ListenFor RequireAppName="ExplicitlySpecified">[Meine] {builtin:AppName} Reise nach {destination} zeigen.</ListenFor>  <Feedback> Zeige Reise nach {destination}</Feedback>  <Navigate Target="foo"/>  </Command>  <Command Name="whenIsTripToDestination">  <Example>Wann ist meine Reise nach Las Vegas?</Example>  <ListenFor RequireAppName="BeforeOrAfterPhrase"> wann ist [meine] Reise nach {destination}</ListenFor>  <ListenFor RequireAppName="ExplicitlySpecified"> wann ist [meine] {builtin:AppName} Reise nach {destination}</ListenFor>  <Feedback> Auf der Suche nach Reise nach {destination}</Feedback>  <VoiceCommandService Target="AdventureWorksVoiceCommandService"/>  </Command>  <Command Name="cancelTripToDestination">  <Example>Meine Reise nach Las Vegas abbrechen</Example>  <ListenFor RequireAppName="BeforeOrAfterPhrase"> [Meine] Reise nach {destination} abbrechen</ListenFor>  <ListenFor RequireAppName="ExplicitlySpecified"> [Meine] {builtin:AppName} Reise nach {destination} abbrechen</ListenFor>  <Feedback> Reise nach {destination} abbrechen</Feedback>  <VoiceCommandService Target="AdventureWorksVoiceCommandService"/>  </Command>  <PhraseList Label="destination">  <Item>London</Item>  <Item>Las Vegas</Item>  <Item>Melbourne</Item>  <Item>Yosemite National Park</Item>  </PhraseList>  </CommandSet>  <CommandSet xml:lang="es-es" Name="AdventureWorksCommandSet\_es-es">  <CommandPrefix> Adventure Works, </CommandPrefix>  <Example>Mostrar viaje a Londres</Example>  <Command Name="showTripToDestination">  <Example> Mostrar viaje a Londres </Example>  <ListenFor RequireAppName="BeforeOrAfterPhrase"> mostrar [mi] viaje a {destination}</ListenFor>  <ListenFor RequireAppName="ExplicitlySpecified"> mostrar [mi] {builtin:AppName} viaje a {destination}</ListenFor>  <Feedback> Mostrando viaje a {destination}</Feedback>  <Navigate Target="foo"/>  </Command>  <Command Name="whenIsTripToDestination">  <Example>¿Cuándo es mi viaje a Las Vegas?</Example>  <ListenFor RequireAppName="BeforeOrAfterPhrase"> cuando es [mi] viaje a {destination}</ListenFor>  <ListenFor RequireAppName="ExplicitlySpecified"> cuando es [mi] {builtin:AppName} viaje a {destination}</ListenFor>  <Feedback>Buscando viaje a {destination}</Feedback>  <VoiceCommandService Target="AdventureWorksVoiceCommandService"/>  </Command>  <Command Name="cancelTripToDestination">  <Example>Cancelar mi viaje a Las Vegas</Example>  <ListenFor RequireAppName="BeforeOrAfterPhrase"> cancelar [mi] viaje a {destination}</ListenFor>  <ListenFor RequireAppName="ExplicitlySpecified"> cancelar [mi] {builtin:AppName} viaje a {destination}</ListenFor>  <Feedback> Viaje a {destination} cancelado </Feedback>  <VoiceCommandService Target="AdventureWorksVoiceCommandService"/>  </Command>  <PhraseList Label="destination">  <Item>London</Item>  <Item>Las Vegas</Item>  <Item>Melbourne</Item>  <Item>Yosemite National Park</Item>  </PhraseList>  </CommandSet>  <CommandSet xml:lang="zh-cn" Name="AdventureWorksCommandSet\_zh-cn">  <CommandPrefix> 奇幻旅行 </CommandPrefix>    <Example>显示伦敦之行</Example>  <Command Name="showTripToDestination">  <Example>显示伦敦之行</Example>  <ListenFor>显示[我]去{destination}的旅行</ListenFor>  <ListenFor RequireAppName="ExplicitlySpecified">显示[我]{builtin:AppName}去{destination}的旅行</ListenFor>  <Feedback>显示{destination}之行</Feedback>  <Navigate Target="foo"/>  </Command>  <Command Name="whenIsTripToDestination">  <Example>我的拉斯维加斯之旅的时间？</Example>  <ListenFor>[我]前往{destination}的旅行是什么时候</ListenFor>  <ListenFor RequireAppName="ExplicitlySpecified">显示[我]{builtin:AppName}前往{destination}的旅行是什么时候</ListenFor>  <Feedback>查询{destination}之行</Feedback>  <VoiceCommandService Target="AdventureWorksVoiceCommandService"/>  </Command>  <Command Name="cancelTripToDestination">  <Example>取消我的拉斯维加斯之旅</Example>  <ListenFor>取消[我的]{destination}之旅</ListenFor>  <ListenFor RequireAppName="ExplicitlySpecified">取消[我的]{builtin:AppName}{destination}之旅</ListenFor>  <Feedback>取消{destination}之行</Feedback>  <VoiceCommandService Target="AdventureWorksVoiceCommandService"/>  </Command>  <PhraseList Label="destination">  <Item>伦敦</Item>  <Item>拉斯维加斯</Item>  <Item>墨尔本</Item>  </PhraseList>  </CommandSet>  </VoiceCommands> |

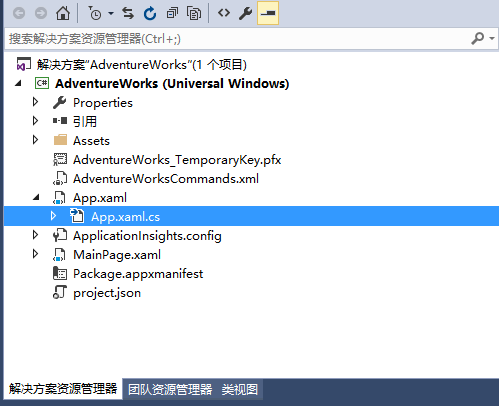
注意：在XML文件中，我们定义了一个中文的语音命令集，在语音命令定义文件中，有一个CommandPrefix节点，它的内容为“奇幻旅行”。这个节点是用户在调用语音命令时使用的前缀。调用语音命令时，用户需要告诉Cortana” 奇幻旅行显示去伦敦的旅行”，即可打开应用程序。按Ctrl + S 保存上面的文件到AdventureWorksCommands.xml中。

**Voice Command Definition（VCD）文件，必须包含如下几个元素**：

* **VoiceCommands 元素：**VCD文件的根节点，xmlns属性必须为“http://schemas.microsoft.com/voicecommands/1.2”，它可以包含1-15个CommandSet节点，每个CommandSet节点代表一个语言的语音命令。
* **CommandSet 元素：**是针对不同语言定义的一组命令。如：xml:lang="zh-cn"，定义了一组中文语音指令。（注意：lang 属性必须是唯一的）
* **CommandPrefix 元素**：是**CommandSet**的子节点，定义了命令的前缀，通常命令组成是以**CommandPrefix**开头**+Command**来实现命令。
* **Command 元素：**是CommandSet的子节点，定义了单独的一个命令。它包含Example(1个)、ListenFor（1到10个）、Feedback（1个）和Navigate（1个）等几个元素，且必须按顺序列出。
* **Example 元素：**告诉用户该App所支持的语音指令的描述。
* **ListenFor 元素：**监听并识别用户的指令，每个Command可以有（1-10个）该元素。（该元素{}中的内容在**PhraseList**节点中可以定义，[]为可以忽略的短语）
* **Feedback 元素：**Cortana识别命令后，给用户的一个反馈。用于Cortana显示。
* **Navigate 元素：**声明语音命令将在前台启动应用。与VoiceCommandService元素互斥，两者只能使用其中一个。
* **VoiceCommandSerice 元素：**声明语音命令将启动后台应用。与Navigate 元素互斥，两者只能使用其中一个。
* **PhraseList 元素：**用来定义一组语音字符，指定相应规定的字符，用来消除歧义, 使用 PhraseList 限制识别适用于一组相对较小的单词。当单词组过大（例如数百个单词）或者根本不应被限制时，要使用 PhraseTopic 元素和 Subject 元素来优化语音识别结果的相关性，从而增强可扩展性。
* **PhraseTopic 元素：**可以提高识别率，内部属性Subject可指定该关键字类型，比如 城市名 姓名 地址 等类型。

### 安装VCD语音命令文件

在解决方案资源管理器中，点击展开App.xaml，打开App.xaml.cs文件。



找到OnLaunched方法，在该方法的最后，添加安装VCD文件的代码，同时还要修改OnLaunched方法为async。安装VCD文件代码如下：

|  |
| --- |
| try  {  StorageFile vcdStorageFile = await Package.Current.InstalledLocation.GetFileAsync(@"AdventureWorksCommands.xml");  await Windows.ApplicationModel.VoiceCommands.VoiceCommandDefinitionManager.InstallCommandDefinitionsFromStorageFileAsync(vcdStorageFile);  }  catch (Exception ex)  {  System.Diagnostics.Debug.WriteLine("Installing Voice Commands Failed: " + ex.ToString());  } |

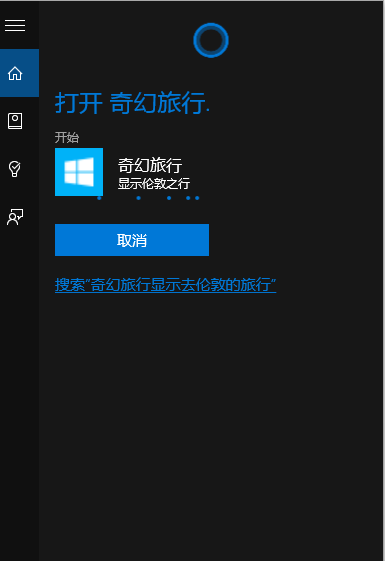
**响应Cortana调用**

我们找到protected override void OnActivated(IActivatedEventArgs args)方法，添加如下代码：

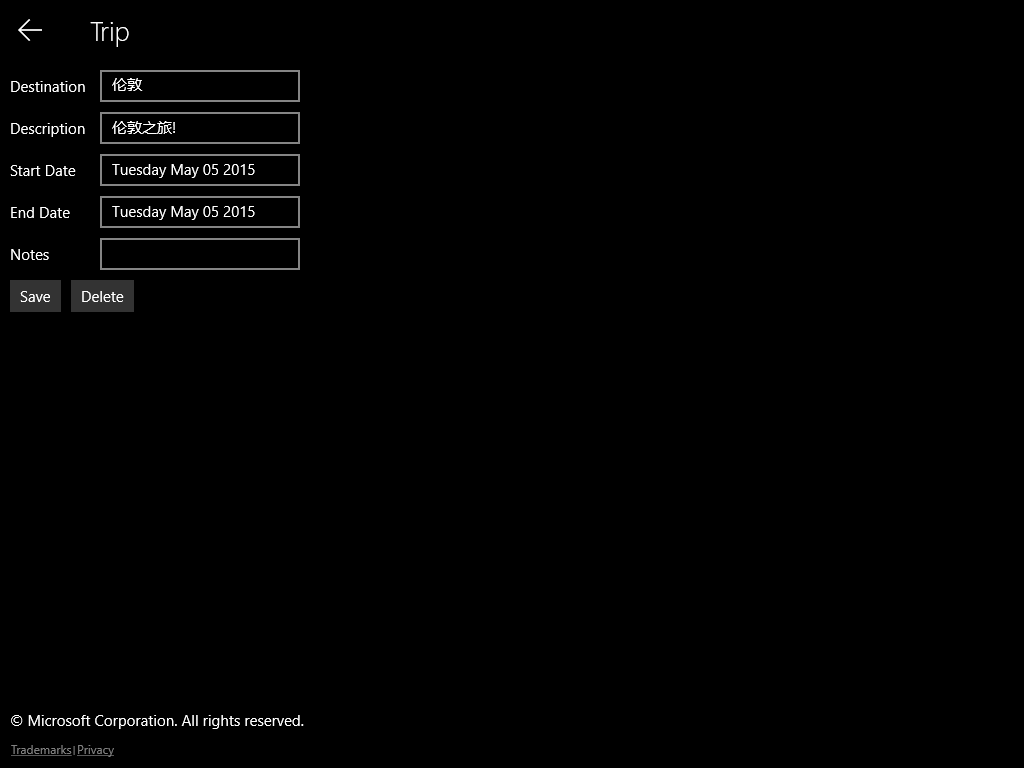
|  |
| --- |
| protected override void OnActivated(IActivatedEventArgs args)  {  base.OnActivated(args);  Type navigationToPageType;  ViewModel.TripVoiceCommand? navigationCommand = null;  // If the app was launched via a Voice Command, this corresponds to the "show trip to <location>" command.  // Protocol activation occurs when a tile is clicked within Cortana (via the background task)  if (args.Kind == ActivationKind.VoiceCommand)  {  // The arguments can represent many different activation types. Cast it so we can get the  // parameters we care about out.  var commandArgs = args as VoiceCommandActivatedEventArgs;  Windows.Media.SpeechRecognition.SpeechRecognitionResult speechRecognitionResult = commandArgs.Result;  // Get the name of the voice command and the text spoken. See AdventureWorksCommands.xml for  // the <Command> tags this can be filled with.  string voiceCommandName = speechRecognitionResult.RulePath[0];  string textSpoken = speechRecognitionResult.Text;  // The commandMode is either "voice" or "text", and it indictes how the voice command  // was entered by the user.  // Apps should respect "text" mode by providing feedback in silent form.  string commandMode = this.SemanticInterpretation("commandMode", speechRecognitionResult);  switch (voiceCommandName)  {  case "showTripToDestination":  // Access the value of the {destination} phrase in the voice command  string destination = this.SemanticInterpretation("destination", speechRecognitionResult);  // Create a navigation command object to pass to the page. Any object can be passed in,  // here we're using a simple struct.  navigationCommand = new ViewModel.TripVoiceCommand(  voiceCommandName,  commandMode,  textSpoken,  destination);  // Set the page to navigate to for this voice command.  navigationToPageType = typeof(View.TripDetails);  break;  default:  // If we can't determine what page to launch, go to the default entry point.  navigationToPageType = typeof(View.TripListView);  break;  }  }  else if (args.Kind == ActivationKind.Protocol)  {  // Extract the launch context. In this case, we're just using the destination from the phrase set (passed  // along in the background task inside Cortana), which makes no attempt to be unique. A unique id or  // identifier is ideal for more complex scenarios. We let the destination page check if the  // destination trip still exists, and navigate back to the trip list if it doesn't.  var commandArgs = args as ProtocolActivatedEventArgs;  Windows.Foundation.WwwFormUrlDecoder decoder = new Windows.Foundation.WwwFormUrlDecoder(commandArgs.Uri.Query);  var destination = decoder.GetFirstValueByName("LaunchContext");  navigationCommand = new ViewModel.TripVoiceCommand(  "protocolLaunch",  "text",  "destination",  destination);  navigationToPageType = typeof(View.TripDetails);  }  else  {  // If we were launched via any other mechanism, fall back to the main page view.  // Otherwise, we'll hang at a splash screen.  navigationToPageType = typeof(View.TripListView);  }  // Re"peat the same basic initialization as OnLaunched() above, taking into account whether  // or not the app is already active.  Frame rootFrame = Window.Current.Content as Frame;  // Do not repeat app initialization when the Window already has content,  // just ensure that the window is active  if (rootFrame == null)  {  // Create a Frame to act as the navigation context and navigate to the first page  rootFrame = new Frame();  App.NavigationService = new NavigationService(rootFrame);  rootFrame.NavigationFailed += OnNavigationFailed;  // Place the frame in the current Window  Window.Current.Content = rootFrame;  }  // Since we're expecting to always show a details page, navigate even if  // a content frame is in place (unlike OnLaunched).  // Navigate to either the main trip list page, or if a valid voice command  // was provided, to the details page for that trip.  rootFrame.Navigate(navigationToPageType, navigationCommand);  // Ensure the current window is active  Window.Current.Activate();  } |

该方法的主要功能是，通过识别Cortana的命令来跳转页面。

运行程序后，VCD文件即可安装注册到Cortana中。VCD文件安装完成后，在Cortana中说出或输入“奇幻旅行显示去伦敦的旅行”，即可打开我们的应用程序。



打开我们的应用程序后，会跳转到“伦敦旅行”的详情页面。



### 在Cortana中使用语音命令启动后台应用

Win 10 SDK给应用程序增加了一个叫做APP Service的技术，应用程序可以通过APP Service公开服务来让其他应用程序调用。APP Service是通过后台任务来处理的，所以不需要我们启动应用程序，调用者只需要知道提供服务的应用程序的程序包名称，以及要调用的服务名称即可进行调用。

### 场景描述

在Cortana中，说出“奇幻旅行前往拉斯维加斯的旅行是什么时候”，Cortana会通过后台App Service，查询出“前往拉斯维加斯”的时间，并显示在Cortana的面板中。

### 添加应用服务代码（APP Service）

打开“解决方案资源管理器”，找到”VoiceCommandService”工程。右键单击，在弹出菜单中选择“添加”->”类”。我们添加一个名为“AdventureWorksVoiceCommandService.cs”的类文件。

在" AdventureWorksVoiceCommandService.cs"文件中，我们需要创建一个实现**IBackgroundTask**接口的新类。**Run**方法是所需要的入口点，在Cortana识别语音命令后调用。（注意：后台任务类本身和后台任务项目中的所有其他类都需要作为密封公共类）。以下是“AdventureWorksVoiceCommandService.cs”文件的代码：

|  |
| --- |
| //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  // Copyright (c) Microsoft. All rights reserved.  // This code is licensed under the MIT License (MIT).  // THIS CODE IS PROVIDED \*AS IS\* WITHOUT WARRANTY OF  // ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING ANY  // IMPLIED WARRANTIES OF FITNESS FOR A PARTICULAR  // PURPOSE, MERCHANTABILITY, OR NON-INFRINGEMENT.  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  using System;  using System.Collections.Generic;  using System.Globalization;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using Windows.ApplicationModel;  using Windows.ApplicationModel.AppService;  using Windows.ApplicationModel.Background;  using Windows.ApplicationModel.Resources.Core;  using Windows.ApplicationModel.VoiceCommands;  using Windows.Storage;  namespace AdventureWorks.VoiceCommands  {  /// <summary>  /// The VoiceCommandService implements the entrypoint for all headless voice commands  /// invoked via Cortana. The individual commands supported are described in the  /// AdventureworksCommands.xml VCD file in the AdventureWorks project. The service  /// entrypoint is defined in the Package Manifest (See section uap:Extension in  /// AdventureWorks:Package.appxmanifest)  /// </summary>  public sealed class AdventureWorksVoiceCommandService : IBackgroundTask  {  /// <summary>  /// the service connection is maintained for the lifetime of a cortana session, once a voice command  /// has been triggered via Cortana.  /// </summary>  VoiceCommandServiceConnection voiceServiceConnection;  /// <summary>  /// Lifetime of the background service is controlled via the BackgroundTaskDeferral object, including  /// registering for cancellation events, signalling end of execution, etc. Cortana may terminate the  /// background service task if it loses focus, or the background task takes too long to provide.  ///  /// Background tasks can run for a maximum of 30 seconds.  /// </summary>  BackgroundTaskDeferral serviceDeferral;  /// <summary>  /// ResourceMap containing localized strings for display in Cortana.  /// </summary>  ResourceMap cortanaResourceMap;  /// <summary>  /// The context for localized strings.  /// </summary>  ResourceContext cortanaContext;  Windows.Globalization.Language cortanaLanguage;//REVERT  VoiceCommandContentTile xyz = null;  /// <summary>  /// Get globalization-aware date formats.  /// </summary>  DateTimeFormatInfo dateFormatInfo;  /// <summary>  /// Background task entrypoint. Voice Commands using the <VoiceCommandService Target="...">  /// tag will invoke this when they are recognized by Cortana, passing along details of the  /// invocation.  ///  /// Background tasks must respond to activation by Cortana within 0.5 seconds, and must  /// report progress to Cortana every 5 seconds (unless Cortana is waiting for user  /// input). There is no execution time limit on the background task managed by Cortana,  /// but developers should use plmdebug (https://msdn.microsoft.com/en-us/library/windows/hardware/jj680085%28v=vs.85%29.aspx)  /// on the Cortana app package in order to prevent Cortana timing out the task during  /// debugging.  ///  /// Cortana dismisses its UI if it loses focus. This will cause it to terminate the background  /// task, even if the background task is being debugged. Use of Remote Debugging is recommended  /// in order to debug background task behaviors. In order to debug background tasks, open the  /// project properties for the app package (not the background task project), and enable  /// Debug -> "Do not launch, but debug my code when it starts". Alternatively, add a long  /// initial progress screen, and attach to the background task process while it executes.  /// </summary>  /// <param name="taskInstance">Connection to the hosting background service process.</param>  public async void Run(IBackgroundTaskInstance taskInstance)  {  serviceDeferral = taskInstance.GetDeferral();  // Register to receive an event if Cortana dismisses the background task. This will  // occur if the task takes too long to respond, or if Cortana's UI is dismissed.  // Any pending operations should be cancelled or waited on to clean up where possible.  taskInstance.Canceled += OnTaskCanceled;  var triggerDetails = taskInstance.TriggerDetails as AppServiceTriggerDetails;  // Load localized resources for strings sent to Cortana to be displayed to the user.  cortanaResourceMap = ResourceManager.Current.MainResourceMap.GetSubtree("Resources");  // Select the system language, which is what Cortana should be running as.  cortanaContext = ResourceContext.GetForViewIndependentUse();  var lang = Windows.Media.SpeechRecognition.SpeechRecognizer.SystemSpeechLanguage.LanguageTag;  cortanaContext.Languages = new string[] { Windows.Media.SpeechRecognition.SpeechRecognizer.SystemSpeechLanguage.LanguageTag };  // Get the currently used system date format  dateFormatInfo = CultureInfo.CurrentCulture.DateTimeFormat;  // This should match the uap:AppService and VoiceCommandService references from the  // package manifest and VCD files, respectively. Make sure we've been launched by  // a Cortana Voice Command.  if (triggerDetails != null && triggerDetails.Name == "AdventureWorksVoiceCommandService")  {  try  {  voiceServiceConnection =  VoiceCommandServiceConnection.FromAppServiceTriggerDetails(  triggerDetails);  voiceServiceConnection.VoiceCommandCompleted += OnVoiceCommandCompleted;  VoiceCommand voiceCommand = await voiceServiceConnection.GetVoiceCommandAsync();  // Depending on the operation (defined in AdventureWorks:AdventureWorksCommands.xml)  // perform the appropriate command.  switch (voiceCommand.CommandName)  {  case "whenIsTripToDestination":  var destination = voiceCommand.Properties["destination"][0];  await SendCompletionMessageForDestination(destination);  break;  case "cancelTripToDestination":  var cancelDestination = voiceCommand.Properties["destination"][0];  await SendCompletionMessageForCancellation(cancelDestination);  break;  default:  // As with app activation VCDs, we need to handle the possibility that  // an app update may remove a voice command that is still registered.  // This can happen if the user hasn't run an app since an update.  LaunchAppInForeground();  break;  }  }  catch (Exception ex)  {  System.Diagnostics.Debug.WriteLine("Handling Voice Command failed " + ex.ToString());  }  }  }  /// <summary>  /// Handle the Trip Cancellation task. This task demonstrates how to prompt a user  /// for confirmation of an operation, show users a progress screen while performing  /// a long-running task, and showing a completion screen.  /// </summary>  /// <param name="destination">The name of a destination, expected to match the phrase list.</param>  /// <returns></returns>  private async Task SendCompletionMessageForCancellation(string destination)  {  // Begin loading data to search for the target store. If this operation is going to take a long time,  // for instance, requiring a response from a remote web service, consider inserting a progress screen  // here, in order to prevent Cortana from timing out.  string progressScreenString = string.Format(  cortanaResourceMap.GetValue("ProgressLookingForTripToDest", cortanaContext).ValueAsString,  destination);  await ShowProgressScreen(progressScreenString);  Model.TripStore store = new Model.TripStore();  await store.LoadTrips();  // We might have multiple trips to the destination. For now, we just pick the first.  IEnumerable<Model.Trip> trips = store.Trips.Where(p => p.Destination == destination);  Model.Trip trip = null;  if (trips.Count() > 1)  {  // If there is more than one trip, provide a disambiguation screen rather than just picking one  // however, more advanced logic here might be ideal (ie, if there's a significant number of items,  // you may want to just fall back to a link to your app where you can provide a deeper search experience.  string disambiguationDestinationString = string.Format(  cortanaResourceMap.GetValue("DisambiguationWhichTripToDest", cortanaContext).ValueAsString,  destination);  string disambiguationRepeatString = cortanaResourceMap.GetValue("DisambiguationRepeat", cortanaContext).ValueAsString;  trip = await DisambiguateTrips(trips, disambiguationDestinationString, disambiguationRepeatString);  }  else  {  // One or no trips exist with that destination, so retrieve it, or return null.  trip = trips.FirstOrDefault();  }  var userPrompt = new VoiceCommandUserMessage();  VoiceCommandResponse response;  if (trip == null)  {  var userMessage = new VoiceCommandUserMessage();  // In this scenario, perhaps someone has modified data on your service outside of this  // apps control. If you're accessing a remote service, having a background task that  // periodically refreshes the phrase list so it's likely to be in sync is ideal.  // This is unlikely to occur for this sample app, however.  string noSuchTripToDestination = string.Format(  cortanaResourceMap.GetValue("NoSuchTripToDestination", cortanaContext).ValueAsString,  destination);  userMessage.DisplayMessage = userMessage.SpokenMessage = noSuchTripToDestination;  response = VoiceCommandResponse.CreateResponse(userMessage);  await voiceServiceConnection.ReportSuccessAsync(response);  }  else  {  // Prompt the user for confirmation that we've selected the correct trip to cancel.  string cancelTripToDestination = string.Format(  cortanaResourceMap.GetValue("CancelTripToDestination", cortanaContext).ValueAsString,  destination);  userPrompt.DisplayMessage = userPrompt.SpokenMessage = cancelTripToDestination;  var userReprompt = new VoiceCommandUserMessage();  string confirmCancelTripToDestination = string.Format(  cortanaResourceMap.GetValue("ConfirmCancelTripToDestination", cortanaContext).ValueAsString,  destination);  userReprompt.DisplayMessage = userReprompt.SpokenMessage = confirmCancelTripToDestination;  //REVERT  var cancelledContentTiles = new List<VoiceCommandContentTile>();  if (xyz != null) cancelledContentTiles.Add(xyz);  response = VoiceCommandResponse.CreateResponseForPrompt(userPrompt, userReprompt, cancelledContentTiles);  var voiceCommandConfirmation = await voiceServiceConnection.RequestConfirmationAsync(response);  // If RequestConfirmationAsync returns null, Cortana's UI has likely been dismissed.  if (voiceCommandConfirmation != null)  {  if (voiceCommandConfirmation.Confirmed == true)  {  string cancellingTripToDestination = string.Format(  cortanaResourceMap.GetValue("CancellingTripToDestination", cortanaContext).ValueAsString,  destination);  await ShowProgressScreen(cancellingTripToDestination);  // Perform the operation to remote the trip from the app's data.  // Since the background task runs within the app package of the installed app,  // we can access local files belonging to the app without issue.  await store.DeleteTrip(trip);  // Provide a completion message to the user.  var userMessage = new VoiceCommandUserMessage();  string cancelledTripToDestination = string.Format(  cortanaResourceMap.GetValue("CancelledTripToDestination", cortanaContext).ValueAsString,  destination);  userMessage.DisplayMessage = userMessage.SpokenMessage = cancelledTripToDestination;  response = VoiceCommandResponse.CreateResponse(userMessage, cancelledContentTiles); //REVERT cancelledContentTiles  response.AppLaunchArgument = destination; //REVERT  await voiceServiceConnection.ReportSuccessAsync(response);  }  else  {  // Confirm no action for the user.  var userMessage = new VoiceCommandUserMessage();  string keepingTripToDestination = string.Format(  cortanaResourceMap.GetValue("KeepingTripToDestination", cortanaContext).ValueAsString,  destination);  userMessage.DisplayMessage = userMessage.SpokenMessage = keepingTripToDestination;  response = VoiceCommandResponse.CreateResponse(userMessage);  response.AppLaunchArgument = destination; //REVERT  await voiceServiceConnection.ReportSuccessAsync(response);  }  }  }  }  /// <summary>  /// Show a progress screen. These should be posted at least every 5 seconds for a  /// long-running operation, such as accessing network resources over a mobile  /// carrier network.  /// </summary>  /// <param name="message">The message to display, relating to the task being performed.</param>  /// <returns></returns>  private async Task ShowProgressScreen(string message)  {  var userProgressMessage = new VoiceCommandUserMessage();  userProgressMessage.DisplayMessage = userProgressMessage.SpokenMessage = message;  VoiceCommandResponse response = VoiceCommandResponse.CreateResponse(userProgressMessage);  await voiceServiceConnection.ReportProgressAsync(response);  }  /// <summary>  /// Demonstrates providing the user with a choice between multiple items. In this case, if a user  /// has two trips to the same destination with different dates, this will provide a way to differentiate  /// them. Provide a way to choose between the items.  /// </summary>  /// <param name="trips">The set of trips to choose between</param>  /// <param name="disambiguationMessage">The initial disambiguation message</param>  /// <param name="secondDisambiguationMessage">Repeat prompt retry message</param>  /// <returns></returns>  private async Task<Model.Trip> DisambiguateTrips(IEnumerable<Model.Trip> trips, string disambiguationMessage, string secondDisambiguationMessage)  {  // Create the first prompt message.  var userPrompt = new VoiceCommandUserMessage();  userPrompt.DisplayMessage =  userPrompt.SpokenMessage = disambiguationMessage;  // Create a re-prompt message if the user responds with an out-of-grammar response.  var userReprompt = new VoiceCommandUserMessage();  userReprompt.DisplayMessage =  userReprompt.SpokenMessage = secondDisambiguationMessage;  // Create items for each item. Ideally, should be limited to a small number of items.  var destinationContentTiles = new List<VoiceCommandContentTile>();  int i = 1;  foreach (Model.Trip trip in trips)  {  var destinationTile = new VoiceCommandContentTile();  // Use a generic background image. This can be fetched from a service call, potentially, but  // be aware of network latencies and ensure Cortana does not time out.  destinationTile.ContentTileType = VoiceCommandContentTileType.TitleWith68x68IconAndText;  destinationTile.Image = await Package.Current.InstalledLocation.GetFileAsync("AdventureWorks.VoiceCommands\\Images\\GreyTile.png");  // The AppContext can be any arbitrary object, and will be maintained for the  // response.  destinationTile.AppContext = trip;  string dateFormat = "";  if (trip.StartDate != null)  {  dateFormat = trip.StartDate.Value.ToString(dateFormatInfo.LongDatePattern);  }  else  {  // The app allows a trip to not have a date, but the choices must be unique  // so they can be spoken aloud and be distinct, so add a number to identify them.  dateFormat = string.Format("{0}", i);  }  //destinationTile.Title = trip.Destination + " " + dateFormat;  //destinationTile.TextLine1 = trip.Description;  destinationTile.Title = trip.Description; //REVERT lines above  destinationTile.TextLine1 = dateFormat;  destinationContentTiles.Add(destinationTile);  i++;  }  // Cortana will handle re-prompting if the user does not provide a valid response.  var response = VoiceCommandResponse.CreateResponseForPrompt(userPrompt, userReprompt, destinationContentTiles);  // If cortana is dismissed in this operation, null will be returned.  var voiceCommandDisambiguationResult = await  voiceServiceConnection.RequestDisambiguationAsync(response);  if (voiceCommandDisambiguationResult != null)  {  xyz = voiceCommandDisambiguationResult.SelectedItem;  return (Model.Trip)voiceCommandDisambiguationResult.SelectedItem.AppContext;  }  return null;  }  /// <summary>  /// Search for, and show details related to a single trip, if the trip can be  /// found. This demonstrates a simple response flow in Cortana.  /// </summary>  /// <param name="destination">The destination, expected to be in the phrase list.</param>  /// <returns></returns>  private async Task SendCompletionMessageForDestination(string destination)  {  // If this operation is expected to take longer than 0.5 seconds, the task must  // provide a progress response to Cortana prior to starting the operation, and  // provide updates at most every 5 seconds.  string loadingTripToDestination = string.Format(  cortanaResourceMap.GetValue("LoadingTripToDestination", cortanaContext).ValueAsString,  destination);  await ShowProgressScreen(loadingTripToDestination);  Model.TripStore store = new Model.TripStore();  await store.LoadTrips();  // Look for the specified trip. The destination \*should\* be pulled from the grammar we  // provided, and the subsequently updated phrase list, so it should be a 1:1 match, including case.  // However, we might have multiple trips to the destination. For now, we just pick the first.  IEnumerable<Model.Trip> trips = store.Trips.Where(p => p.Destination == destination);  var userMessage = new VoiceCommandUserMessage();  var destinationsContentTiles = new List<VoiceCommandContentTile>();  if (trips.Count() == 0)  {  // In this scenario, perhaps someone has modified data on your service outside of your  // control. If you're accessing a remote service, having a background task that  // periodically refreshes the phrase list so it's likely to be in sync is ideal.  // This is unlikely to occur for this sample app, however.  string foundNoTripToDestination = string.Format(  cortanaResourceMap.GetValue("FoundNoTripToDestination", cortanaContext).ValueAsString,  destination);  userMessage.DisplayMessage = foundNoTripToDestination;  userMessage.SpokenMessage = foundNoTripToDestination;  }  else  {  // Set a title message for the page.  string message = "";  if (trips.Count() > 1)  {  message = cortanaResourceMap.GetValue("PluralUpcomingTrips", cortanaContext).ValueAsString;  }  else  {  message = cortanaResourceMap.GetValue("SingularUpcomingTrip", cortanaContext).ValueAsString;  }  userMessage.DisplayMessage = message;  userMessage.SpokenMessage = message;  // file in tiles for each destination, to display information about the trips without  // launching the app.  foreach (Model.Trip trip in trips)  {  int i = 1;  var destinationTile = new VoiceCommandContentTile();  destinationTile.ContentTileType = VoiceCommandContentTileType.TitleWith68x68IconAndText;  destinationTile.Image = await Package.Current.InstalledLocation.GetFileAsync("AdventureWorks.VoiceCommands\\Images\\GreyTile.png");  destinationTile.AppLaunchArgument = string.Format("destination={0}", trip.Destination);  destinationTile.Title = trip.Description; //.Destination - REVERT  if (trip.StartDate != null)  {  destinationTile.TextLine1 = trip.StartDate.Value.ToString(dateFormatInfo.LongDatePattern);  }  else  {  destinationTile.TextLine1 = trip.Destination + " " + i;  }  //REVERT  destinationTile.AppLaunchArgument = destination;  destinationsContentTiles.Add(destinationTile);  i++;  }  }  var response = VoiceCommandResponse.CreateResponse(userMessage, destinationsContentTiles);  if (trips.Count() > 0)  {  //REVERT  //response.AppLaunchArgument = string.Format("destination={0}", destination);  response.AppLaunchArgument = string.Format(destination);  }  await voiceServiceConnection.ReportSuccessAsync(response);  }  /// <summary>  /// Provide a simple response that launches the app. Expected to be used in the  /// case where the voice command could not be recognized (eg, a VCD/code mismatch.)  /// </summary>  private async void LaunchAppInForeground()  {  var userMessage = new VoiceCommandUserMessage();  userMessage.SpokenMessage = cortanaResourceMap.GetValue("LaunchingAdventureWorks", cortanaContext).ValueAsString;  var response = VoiceCommandResponse.CreateResponse(userMessage);  response.AppLaunchArgument = "";  await voiceServiceConnection.RequestAppLaunchAsync(response);  }  /// <summary>  /// Handle the completion of the voice command. Your app may be cancelled  /// for a variety of reasons, such as user cancellation or not providing  /// progress to Cortana in a timely fashion. Clean up any pending long-running  /// operations (eg, network requests).  /// </summary>  /// <param name="sender">The voice connection associated with the command.</param>  /// <param name="args">Contains an Enumeration indicating why the command was terminated.</param>  private void OnVoiceCommandCompleted(VoiceCommandServiceConnection sender, VoiceCommandCompletedEventArgs args)  {  if (this.serviceDeferral != null)  {  this.serviceDeferral.Complete();  }  }  /// <summary>  /// When the background task is cancelled, clean up/cancel any ongoing long-running operations.  /// This cancellation notice may not be due to Cortana directly. The voice command connection will  /// typically already be destroyed by this point and should not be expected to be active.  /// </summary>  /// <param name="sender">This background task instance</param>  /// <param name="reason">Contains an enumeration with the reason for task cancellation</param>  private void OnTaskCanceled(IBackgroundTaskInstance sender, BackgroundTaskCancellationReason reason)  {  System.Diagnostics.Debug.WriteLine("Task cancelled, clean up");  if (this.serviceDeferral != null)  {  //Complete the service deferral  this.serviceDeferral.Complete();  }  }  }  } |

* **public async void Run(IBackgroundTaskInstance taskInstance)：**后台服务的入口点，当后台服务启动时，会调用该方法。

**taskInstance 参数：**连接后台服务的进程实例。

* **private async Task SendCompletionMessageForCancellation(string destination):**该方法用于处理取消旅行行程的操作任务，该方法演示了如何提示用户在与Cortana交互过程中的确认操作与进度显示。

**Destination 参数：**目的地的名称，用来匹配短语列表。

* **private async Task ShowProgressScreen(string message)：**该方法用于在Cortana面板中显示提示语与进度。

**message 参数：**在Cortana面板中显示的消息。

* **private async Task<Model.Trip> DisambiguateTrips(IEnumerable<Model.Trip> trips, string disambiguationMessage, string secondDisambiguationMessage)：**该方法用于，为用户提供多个项目之间的选择。如果用户有到同一目的地的不同日期，该方法将提供一种方式来区分他们。

**trips 参数：**设定的旅行选择。

**disambiguationMessage 参数：**提示消息。

**secondDisambiguationMessage 参数：**从试提示消息。

* **private async Task SendCompletionMessageForDestination(string destination)：**该方法用于搜索目的地并显示目的地详情。

**destination 参数：**目的地名称。

* **private async void LaunchAppInForeground()：**该方法用于启动应用程序。
* **private void OnVoiceCommandCompleted(VoiceCommandServiceConnection sender, VoiceCommandCompletedEventArgs args)：**该方法用于取消已经完成的语音命令。
* **private void OnTaskCanceled(IBackgroundTaskInstance sender, BackgroundTaskCancellationReason reason)：**该方法用于取消任务。

### 知识点

后台任务的功能主要是响应Cortana收到的语音命令，应用程序通过后台App Service来触发，这样我们就能在后台任务中进行交互。

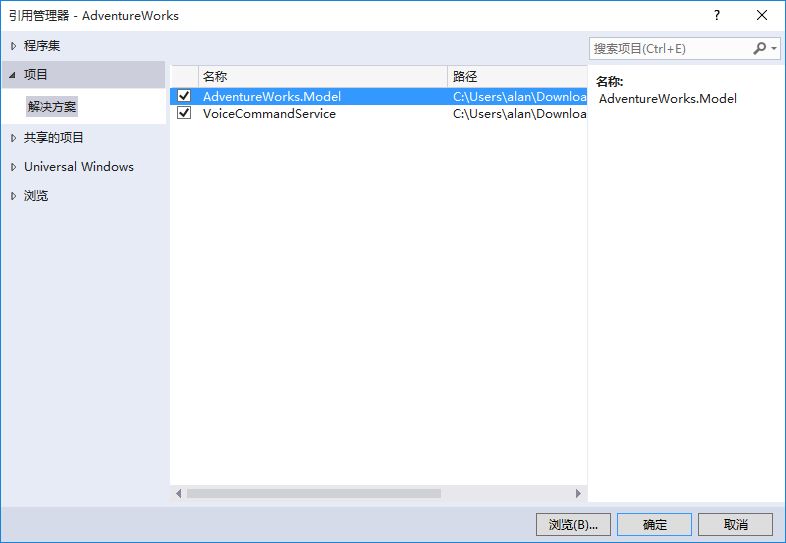
**VoiceCommandServiceConnection类**：它的实例可以从后台任务实例的触发器数据中获得。它使用“FromAppServiceTriggerDetails”方法来获取实例。该类有几个重要的方法如下：

* **ReportSuccessAsync(VoiceCommandResponse response)**：通知Cortana，处理已经完成，并返回一条消息，传递给Cortana的面板显示。（注意：一旦Cortana启动后台服务，应用服务有0.5秒的时间调用ReportSuccessAsync方法。Cortana使用应用提供的数据显示并说出VCD文件中指定的反馈。如果应用进行该调用的时间超过0.5秒，Cortana将显示切换屏幕，直到应用程序调用ReportSuccessAsync方法为止，最多显示5秒。如果应用服务不调用 **ReportSuccessAsync**或任何提供为 **Cortana** 提供信息的 [**VoiceCommandServiceConnection**](https://msdn.microsoft.com/zh-cn/library/windows.applicationmodel.voicecommands.voicecommandserviceconnection.aspx) 方法，则用户将收到一条错误消息并且该应用服务将会取消。）
* **ReportFailureAsync(VoiceCommandResponse response)**：该方法用于向Cortana反馈错误信息。
* **ReportProgressAsync(VoiceCommandResponse response)**：该方法用于向Cortana报告进度，但并不指定具体的进度值，只是在Cortana面板上会显示时长5秒钟的进度条。
* **RequestAppLaunchAsync(VoiceCommandResponse response)**：该方法用于向Cortana请求启动当前应用。
* **RequestConfirmationAsync(VoiceCommandResponse response)**：该方法用于向Cortana面板发送一条需要用户确认的消息，后台服务会等待用户的确认结果，以决定下一步做什么操作。
* **RequestDisambiguationAsync(VoiceCommandResponse response)**：该方法用于向Cortana面板发送一条需要用户确认的消息。与上面方法不同的是，这个方法会在Cortana面板上列出多个选项，让用户说出或点击要选择的选项，Cortana会把用户选择的项返回给应用服务，以决定下一步做什么操作。（注意：作为方法参数的VoiceCommandResponse实例一定要使用CreateResponseForPrompt静态方法来创建）

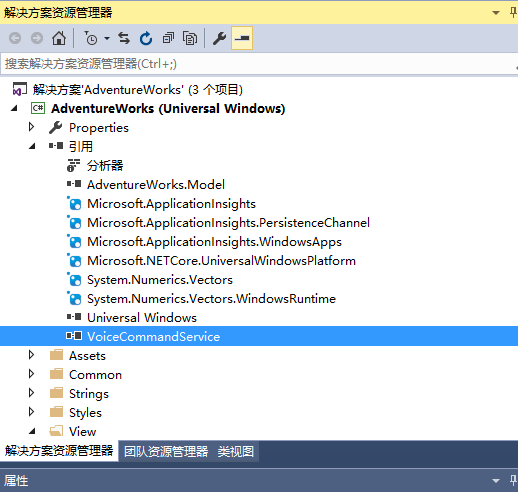
**VoiceCommandUserMessage 类：**该类主要用于Cortana在与后台服务的交互过程中，向Cortana面板发送消息的封装。它有两个重要的属性：

* **DisplayMessage:**要显示在Cortana面板上的文本信息。
* **SpokenMessage:**希望Cortana说出来的文本。

**添加后台服务引用：**在“解决方案资源管理器“中，找到AdventureWorks工程，右键单击”引用“，在弹出菜单中选择”添加引用“。



找到项目->解决方案->VoiceCommandService项目，勾选后点击“确定“，即可将后台服务添加到当前项目中。



### 配置App Service

在“解决方案资源管理器”中，右键单击 "Package.appxmanifest" 文件并选择“查看代码”。找到Application元素，在该元素下添加”Extension 元素”,配置项如下：

|  |
| --- |
| <Package>  <Applications>  <Application>  <Extensions>  <uap:Extension Category="windows.appService" EntryPoint="AdventureWorks.VoiceCommands.AdventureWorksVoiceCommandService">  <uap:AppService Name="AdventureWorksVoiceCommandService" />  </uap:Extension>  </Extensions>  <Application>  <Applications>  </Package> |

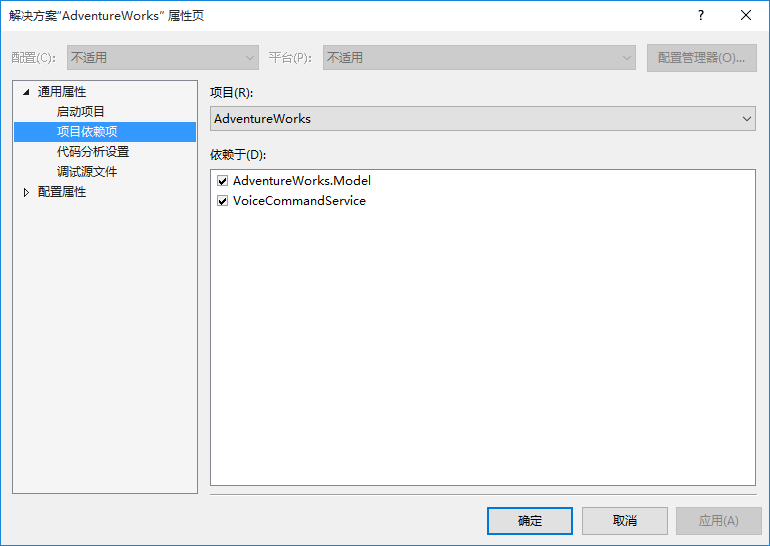
在Extension元素中，有两个重要的属性，Category和EntryPoint。

**Category 属性**：定义扩展的类型，此处为一个APP Service.

**EntryPoint 属性**：后台应用服务的一个入口命名空间，我们在这里指定我们定义好的类“AdventureWorksVoiceCommandService”。当Cortana调用后台服务时，会运行该类的Run方法。

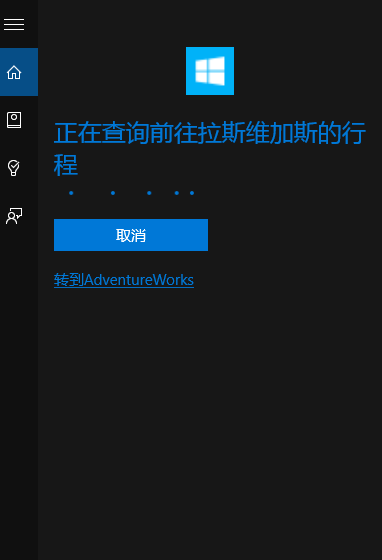
**AppService 元素**：定义了App Service的名称。

**设置解决方案的项目依赖：**在“解决方案资源管理器“中，右键点击解决方案”AdventureWorks“，在弹出菜单中点击”属性“。

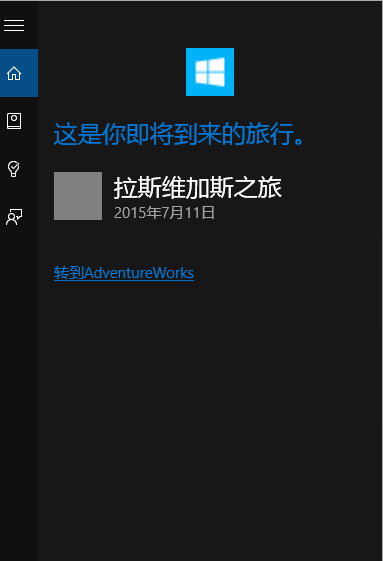


点击展开通用属性，选择项目依赖项，勾选VoiceCommandService点击确定。

运行一次应用程序。在Cortana中说出或敲入“奇幻旅行前往拉斯维加斯的旅行是什么时候”。



Cortana与后台服务交互完成后，即可在Cortana面板中查看“前往拉斯维加斯”时间。



### 动态修改语音命令定义（VCD）短语列表

修改语音命令定义（VCD）短语列表，实际上是修改VCD文件中的PhraseList元素中的Item项。有的时候，为了多语言语音命令的支持，我们需要动态的修改PhraseList内容，以适应多语言。如：在设备当前语言为英语时，PraseList中的内容为{London，Las Vegas，Melbourne，Yosemite National Park}。在Command中，如果短语命中ParseList中的项目时，Cortana会根据命令来调用前台程序或后台服务。但当我们切换当前设备语言为中文时，我们需要更新PraseList中的内容为{伦敦，拉斯维加斯，墨尔本，约塞米蒂国家公园}，以支持中文的语音命令。（**在开始实践之前，我们建议先卸载已经安装的AdventureWorks,为了保证动态短语的数据完整性，我们需要重新安装VCD文件**）

打开“解决方案资源管理器”中的“AdventureWorksCommands.xml”文件，修改"AdventureWorksCommandSet\_zh-cn"中文节点的PhraseList元素为如下代码：

|  |
| --- |
| <PhraseList Label="destination">  <Item>London</Item>  <Item>Las Vegas</Item>  <Item>Melbourne</Item>  <Item>Yosemite National Park</Item>  </PhraseList> |

在“解决方案资源管理器中”，单击打开App.xaml.cs文件找到OnLaunched方法，添加更新短语代码，OnLaunched方法完成代码如下：

|  |
| --- |
| /// <summary>  /// Invoked when the application is launched normally by the end user. Other entry points  /// will be used such as when the application is launched to respond to voice commands, or  /// URI invoked. This also installs the voice commands into cortana.  /// </summary>  /// <param name="e">Details about the launch request and process.</param>  protected async override void OnLaunched(LaunchActivatedEventArgs e)  {  Frame rootFrame = Window.Current.Content as Frame;  // Do not repeat app initialization when the Window already has content,  // just ensure that the window is active  if (rootFrame == null)  {  // Create a Frame to act as the navigation context and navigate to the first page  rootFrame = new Frame();  App.NavigationService = new NavigationService(rootFrame);  rootFrame.NavigationFailed += OnNavigationFailed;  // Place the frame in the current Window  Window.Current.Content = rootFrame;  }  if (rootFrame.Content == null)  {  // Determine if we're being activated normally, or with arguments from Cortana.  if (string.IsNullOrEmpty(e.Arguments))  {  // Launching normally.  rootFrame.Navigate(typeof(View.TripListView), "");  }  else  {  // Launching with arguments. We assume, for now, that this is likely  // to be in the form of "destination=<location>" from activation via Cortana.  rootFrame.Navigate(typeof(View.TripDetails), e.Arguments);  }  }  // Ensure the current window is active  Window.Current.Activate();  try  {  // Install the main VCD. Since there's no simple way to test that the VCD has been imported, or that it's your most recent  // version, it's not unreasonable to do this upon app load.  StorageFile vcdStorageFile = await Package.Current.InstalledLocation.GetFileAsync(@"AdventureWorksCommands.xml");  await Windows.ApplicationModel.VoiceCommands.VoiceCommandDefinitionManager.InstallCommandDefinitionsFromStorageFileAsync(vcdStorageFile);  //Update phrase list.  ViewModel.ViewModelLocator locator = App.Current.Resources["ViewModelLocator"] as ViewModel.ViewModelLocator;  if (locator != null)  {  await locator.TripViewModel.UpdateDestinationPhraseList();  }  }  catch (Exception ex)  {  System.Diagnostics.Debug.WriteLine("Installing Voice Commands Failed: " + ex.ToString());  }  } |

**VoiceCommandDefinitionManager 类**：该类用于VCD文件的管理。

* **InstallCommandDefinitionsFromStorageFileAsync(StorageFile file) 方法**：该方法用于安装注册VCD文件。
* **InstalledCommandDefinitions 属性**：该属性返回VCD文件的描述。

VoiceCommandDefinition类中包含一个重要的方法SetPhraseListAsync，该方法用于更新VCD文件中的PhraseList元素。（注意：如果你使用该方法修改短语列表，将替换整个短语列表）

运行一次程序，在Cortana说出或输入“奇幻旅行显示去伦敦的旅行”会打开我们的应用程序，并跳转到伦敦旅行的详情页面。

