Windows Phone Mango编程实践

***Windows Phone Mango Programming Practice***

第二篇 Silverlight交互篇

# 推送通知(Push Notifications)

## 概述

假设我们设计开发旅行信息服务应用程序，该应用程序提供有关用户选择的航班的信息。当用户将旅行信息输入到应用程序时，该信息将上载到Web Service，Web Service将不断轮询提供其航班和天气数据等信息。

当航班状态信息或者天气情况发生变化，需要尽可能迅速高效地将该信息提供给用户。为此，一种方法移动应用程序经常主动与Web Service通讯，以了解是否有任何等待处理的通知。这样做虽然有效，但是会导致手机的无线设备频繁打开，从而对电池续航时间带来负面影响；另一种方法是让该服务将信息推送至客户端应用程序。这将在数据变得可用时为用户提供对最新可用数据集的访问。由于将数据推送至了客户端，因此即使用户丢失其网络连接，数据也可用。显而易见，后者的方法是受推崇的。

我们可借助Windows Azure服务通过使用Windows Phone推送通知实现这一点。

微软推送通知服务（Microsoft Push Notification Service）是 Microsoft 托管服务的一部分，用于将消息中继到 Windows Phone设备，该服务可供所有 Windows Phone应用程序开发人员使用。使用推送通知的方式取代主动查找，Web Service能够告知应用程序及时获取所需要的重要更新。

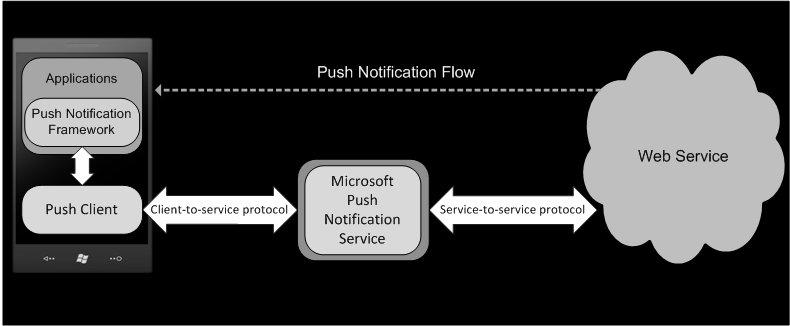
* 1. 

图7-1 推送通知

## 推送通知工作原理

Windows Phone提供的推送通知服务，支持第三方Web服务以高效的方式将数据由专用的通道发送到 Windows Phone应用程序。

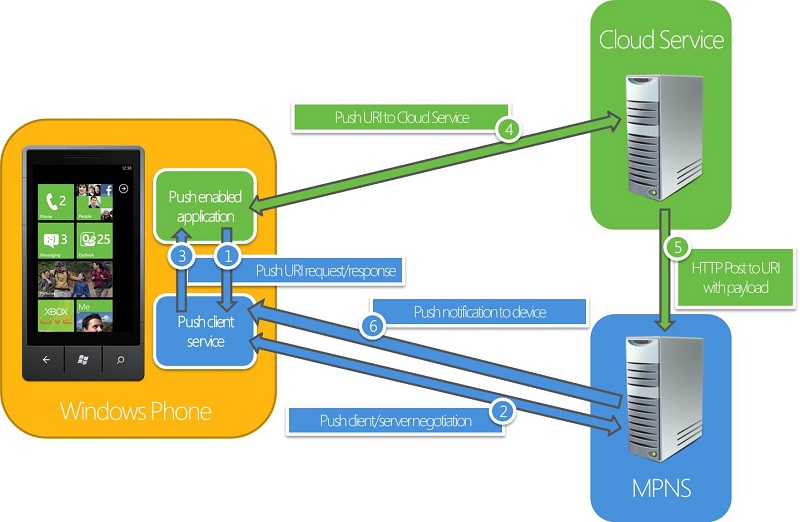


图7-2 工作原理

1. 智能终端应用程序向推送客户端服务请求推送通知的URI；

2、3. 推送客户端服务与微软推送通知服务（MPNS）通讯后，向智能终端应用程序返回推送通知的URI；

4. 智能终端应用程序发送推送通知的URI至Web或者云服务；

5. Web或者云服务使用推送通知的URI向微软推送通知服务（MPNS）发送推送通知信息；

6. 微软推送通知服务（MPNS）将推送通知消息发送给智能终端应用程序。

## 推送通知类型

### Toast通知

Toast通知是在屏幕的最上方显示通知事件信息，比如新闻或天气提醒。Toast通知显示时间约 10 秒钟，用户在Toast通知消息上向右轻划手指也可以使Toas通知消失。如果用户点击Toast通知，将启动接收Toast通知的应用程序。

Toast通知包含的内容：

* **标题**。在应用程序图标后面显示的粗体字符串。XML 架构中的Text1属性定义的就是标题。
* **副标题**。在**标题**后面显示的非粗体字符串。XML 架构中的Text2属性定义的就是副标题。
* **参数**。不显示，如果用户点击Toast通知，此参数可以指示应用程序应启动到哪一页面，它还可以包含要传递给应用程序的值。在 XML 架构中的Param属性定义的就是参数。

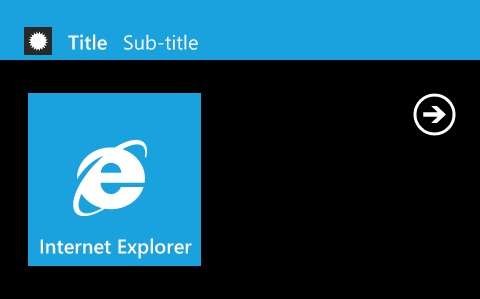


图7-3 Toast通知

在Toast通知中Windows Phone显示微缩的应用程序图标，在应用程序编写时，开发者可以修改应用程序关联的图标，以便在推送通知时区别于其他应用程序。

### Tile通知

每个应用程序可设置Tile—应用程序内容的可视化、 动态的表示形式。当应用程序被固定显示在启动屏幕(Start Screen)时，我们就可以看到Tile的信息。与Windows Phone OS 7.0不同的是，Mango的Tile通知增加了BackBackgroundImage、BackTitle和BackContent。

下图所示的是显示在Tile通知前面的消息内容。

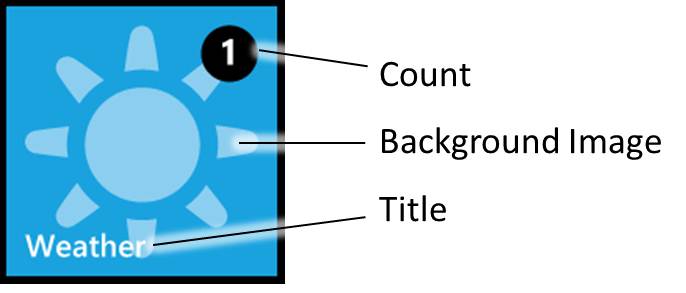


图7-4 Tile通知（Front）

* 背景图片。背景图片可以使用本地资源或远程资源的图片。如果要使用本地资源，它必须已安装的 XAP包的一部分或保存在独立存储中图片。由于网络在可变性和稳定性方面的原因，建议背景图片使用本地资源的图片。在Tile通知中请始终设置背景图像的属性。
* 标题。指示应用程序Tile通知的的标题。标题必须是单行文本，而且长度不应大于实际Tile通知的宽度。如果设置为空字符串，标题将不会显示。
* 计数。1 到 99的整数值。如果计数值没有设置或者设置为0，那么显示计数的圆和计数值值将不会显示。

下图所示的是显示在Tile通知背面的消息内容。

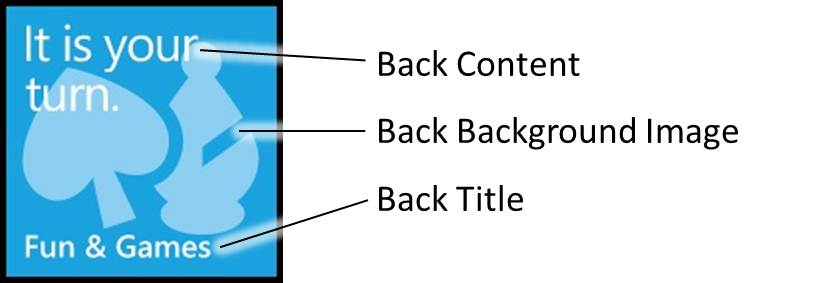


图7-5 Tile通知（Back）

* BackBackgroundImage。背面的背景图像可以使用本地资源或远程资源。如果要使用本地资源，它必须已安装的 XAP包的一部分或保存在独立存储中图片。由于网络在可变性和稳定性方面的原因，建议背面的背景图片使用本地资源的图片。如果设置资源文件的URI为空，那么BackBackgroundImage将不会显示。
* BackTitle。背面Tile通知的底部显示的字符串。如果设置该字符串为空，BackTitle将不会显示。
* BackContent。背面Tile通知的正文中显示的字符串。如果设置字符串为空，BackContent将不会显示。

### Raw通知

Raw通知的格式可以任意设定。如果当前没有运行您的应用程序，Raw通知将被微软推通知服务丢弃，不会传递到Windows Phone设备。

### [[http://i.msdn.microsoft.com/Hash/030c41d9079671d09a62d8e2c1db6973.gif](javascript:void(0))推送通知类型选择](javascript:void(0))

通知是用户体验中的重要组成部分，您需要仔细考虑它的使用方式。重复通知或侵入式通知会降低您的应用程序及设备上运行的其他程序的性能。这些通知还会打扰用户。

请考虑发送通知的频率以及您希望引起用户注意的事件类型。

|  |  |
| --- | --- |
| 推式通知类型 | 应用示例 |
| Tile通知 | 如天气应用程序中的温度变化的信息性通知。 |
| Toast通知 | 立即查看，如突发新闻的重要通知。 |
| Raw通知 | 以自定义的格式将信息直接发送到您的应用程序。 |

表7-1 推送通知类型

## 动手实践——sub-tiles and deep toast通知（Windows Phone Mango）

本节示例的Windows Phone应用程序能够单独接收来自每个城市的天气信息，用户选择城市并将该城市的sub-tile天气信息固定显示在启动屏幕（Start Screen）。

本节中我们将使用 sub-tiles在启动屏幕（Start Screen）中显示有关特定城市的天气信息。单击sub-tile将打开天气预报的应用程序，并将显示该位置的天气信息。我们也将接收来自微软推送通知服务（MNPS）的新的Toast通知，单击新的Toast通知，系统将导航到显示特定的城市的天气信息的页面。本节所讲述的技术是Windows Phone Mango新增的功能，与之前的OS 7.0有很大的改进。

### 前提

本节的动手实践——sub-tiles and deep toast引用了WindowsPhone.Recipes.Push.Messasges.dll，因此需要下载Windows Azure Toolkit for Windows Phone工具包。

或者您也可以使用本章的代码文件PushNotificationsMango\Assets\lib里的WindowsPhone.Recipes.Push.Messasges.dll。更多详细的信息请登录<http://watoolkitwp7.codeplex.com/> 。

安装[**Silverlight for Windows Phone Toolkit**](http://silverlight.codeplex.com/)，下载地址<http://silverlight.codeplex.com/> ，如图7-6 Silverlight for Windows Phone Toolkit。





图7-6 Silverlight for Windows Phone Toolkit

### Sub-Tiles

1. 微软推送通知服务（MPNS）注册

本节我们将实现为应用程序中的每个页面都在微软推送通知服务（MPNS）注册，确保所有的应用程序的功能独立于任何特定的页面。要知道，在这之前我们只能为应用程序的主页面在MSPN注册。

1. 以管理员身份启动Microsoft Visual Studio 2010 Express for Windows® Phone ，选择**开始|所有程序 | Microsoft Visual Studio 2010 Express | Microsoft Visual Studio 2010 Express for Windows® Phone**。

Visual Studio 2010: 以管理员身份启动Visual Studio 2010，选择**开始 | 所有程序 | Microsoft Visual Studio 2010**。

**注意:**

以管理员身份Microsoft Visual Studio 2010 Express for Windows® Phone的设置方法。选择开始|所有程序 | Microsoft Visual Studio 2010 Express，或者**开始 | 所有程序 | All Programs | Microsoft Visual Studio 2010**，点击右键后选择“以管理员身份运行”。在弹出的用户账号控制窗体中选择“是”。

1. 打开LocationInformation.cs文件， LocationInformation类添加的引用如下。

Solution: PushNotificationsMango Project: WPPushNotification.TestClient

File: LocationInformation.cs

using System;

using System.ComponentModel;

1. LocationInformation类继承于**INotifyPropertyChanged** ，此类封装与 sub-tile 相关的位置有关的所有数据：应用程序是否固定显示在启动屏幕、地理位置名称和温度。

Solution: PushNotificationsMango Project: WPPushNotification.TestClient

File: LocationInformation.cs

public class LocationInformation : INotifyPropertyChanged

{

private bool tilePinned;

private string name;

private string temperature;

private string imageName;

/// <summary>

/// Whether or not the location's secondary tile has been

/// pinned by the user.

/// </summary>

public bool TilePinned

{

get

{

return tilePinned;

}

set

{

if (value != tilePinned)

{

tilePinned = value;

if (PropertyChanged != null)

{

PropertyChanged(this, new PropertyChangedEventArgs("TilePinned"));

}

}

}

}

/// <summary>

/// The location's name.

/// </summary>

public string Name

{

get

{

return name;

}

set

{

if (value != name)

{

name = value;

if (PropertyChanged != null)

{

PropertyChanged(this, new PropertyChangedEventArgs("Name"));

}

}

}

}

/// <summary>

/// The temperature at the location.

/// </summary>

public string Temperature

{

get

{

return temperature;

}

set

{

if (value != temperature)

{

temperature = value;

if (PropertyChanged != null)

{

PropertyChanged(this, new PropertyChangedEventArgs("Temperature"));

}

}

}

}

/// <summary>

/// The name of the image to use for representing the weather

/// at the location.

/// </summary>

public string ImageName

{

get

{

return imageName;

}

set

{

if (value != imageName)

{

imageName = value;

if (PropertyChanged != null)

{

PropertyChanged(this, new PropertyChangedEventArgs("ImageName"));

}

}

}

}

public event PropertyChangedEventHandler PropertyChanged;

}

1. **WPPushNotification.TestClient**的**Status**类继承于**INotifyPropertyChanged**，支持文本消息的更改通知。

Solution: PushNotificationsMango Project: WPPushNotification.TestClient

File: Status.cs

public class Status : INotifyPropertyChanged

{

private string message;

/// <summary>

/// A message representing some status.

/// </summary>

public string Message

{

get

{

return message;

}

set

{

if (value != message)

{

message = value;

if (PropertyChanged != null)

{

PropertyChanged(this, new PropertyChangedEventArgs("Message"));

}

}

}

}

#region INotifyPropertyChanged Members

public event PropertyChangedEventHandler PropertyChanged;

#endregion

}

1. **WPPushNotification.TestClient**工程的**PushHandler**类建立与MSPND的连接。我们将使用Locations属性映射每个位置的信息。

Solution: PushNotificationsMango Project: WPPushNotification.TestClient

File: PushHandler.cs

private HttpNotificationChannel httpChannel;

const string channelName = "WeatherUpdatesChannel";

private bool connectedToMSPN;

private bool connectedToServer;

private bool notificationsBound;

/// <summary>

/// Contains information about the locations displayed by the application.

/// </summary>

public Dictionary<string, LocationInformation> Locations { get; private set; }

/// <summary>

/// A dispatcher used to interact with the UI.

/// </summary>

public Dispatcher Dispatcher { get; private set; }

/// <summary>

/// Push service related status information.

/// </summary>

public Status PushStatus { get; private set; }

/// <summary>

/// Whether or not the handler has fully established a connection to both the MSPN and the application server.

/// </summary>

public bool ConnectionEstablished

{

get

{

return connectedToMSPN && connectedToServer && notificationsBound;

}

}

1. PushHandler类的构造函数，初始化类的属性。

Solution: PushNotificationsMango Project: WPPushNotification.TestClient

File: PushHandler.cs

public PushHandler (Status pushStatus, Dictionary<string, LocationInformation> locationsInformation,

Dispatcher uiDispatcher)

{

PushStatus = pushStatus;

Locations = locationsInformation;

Dispatcher = uiDispatcher;

}

1. EstablishConnections方法中建立了所有必要的连接，包括与MSPN的连接和WPF的服务器应用程序的连接。

Solution: PushNotificationsMango Project: WPPushNotification.TestClient

File: PushHandler.cs

/// <summary>

/// Connects to the Microsoft Push Service and registers the received channel with the application server.

/// </summary>

public void EstablishConnections()

{

connectedToMSPN = false;

connectedToServer = false;

notificationsBound = false;

try

{

//First, try to pick up existing channel

httpChannel = HttpNotificationChannel.Find(channelName);

if (null != httpChannel)

{

connectedToMSPN = true;

App.Trace("Channel Exists - no need to create a new one");

SubscribeToChannelEvents();

App.Trace("Register the URI with 3rd party web service");

SubscribeToService();

App.Trace("Subscribe to the channel to Tile and Toast notifications");

SubscribeToNotifications();

UpdateStatus("Channel recovered");

}

else

{

App.Trace("Trying to create a new channel...");

//Create the channel

httpChannel = new HttpNotificationChannel(channelName, "HOLWeatherService");

App.Trace("New Push Notification channel created successfully");

SubscribeToChannelEvents();

App.Trace("Trying to open the channel");

httpChannel.Open();

UpdateStatus("Channel open requested");

}

}

catch (Exception ex)

{

UpdateStatus("Channel error: " + ex.Message);

}

}

1. SubscribeToChannelEvents 方法实现MSPN的通道相关的事件的注册；SubscribeToService 方法实现WPF服务器注册；SubscribeToNotifications方法实现绑定Toast和Tile通知。

Solution: PushNotificationsMango Project: WPPushNotification.TestClient

File: PushHandler.cs

private void SubscribeToChannelEvents()

{

//Register to UriUpdated event - occurs when channel successfully opens

httpChannel.ChannelUriUpdated += new EventHandler<NotificationChannelUriEventArgs>(httpChannel\_ChannelUriUpdated);

//Subscribed to Raw Notification

httpChannel.HttpNotificationReceived += new EventHandler<HttpNotificationEventArgs>(httpChannel\_HttpNotificationReceived);

//general error handling for push channel

httpChannel.ErrorOccurred += new EventHandler<NotificationChannelErrorEventArgs>(httpChannel\_ExceptionOccurred);

//subscribe to toast notification when running app

httpChannel.ShellToastNotificationReceived += new EventHandler<NotificationEventArgs>(httpChannel\_ShellToastNotificationReceived);

}

private void SubscribeToService()

{

//Hardcode for solution - need to be updated in case the REST WCF service address change

string baseUri = "http://localhost:8000/RegirstatorService/Register?uri={0}";

string theUri = String.Format(baseUri, httpChannel.ChannelUri.ToString());

WebClient client = new WebClient();

client.DownloadStringCompleted += (s, e) =>

{

if (null == e.Error)

{

connectedToServer = true;

UpdateStatus("Registration succeeded");

}

else

{

UpdateStatus("Registration failed: " + e.Error.Message);

}

};

client.DownloadStringAsync(new Uri(theUri));

}

private void SubscribeToNotifications()

{

//////////////////////////////////////////

// Bind to Toast Notification

//////////////////////////////////////////

try

{

if (httpChannel.IsShellToastBound == true)

{

App.Trace("Already bound to Toast notification");

}

else

{

App.Trace("Registering to Toast Notifications");

httpChannel.BindToShellToast();

}

}

catch (Exception ex)

{

// handle error here

App.Trace("Bind to Toast Notification Exception : " + ex.Message);

throw ex;

}

//////////////////////////////////////////

// Bind to Tile Notification

//////////////////////////////////////////

try

{

if (httpChannel.IsShellTileBound == true)

{

App.Trace("Already bound to Tile Notifications");

}

else

{

App.Trace("Registering to Tile Notifications");

// you can register the phone application to receive tile images from remote servers [this is optional]

Collection<Uri> uris = new Collection<Uri>();

uris.Add(new Uri("http://www.larvalabs.com"));

httpChannel.BindToShellTile(uris);

}

}

catch (Exception ex)

{

//handle error here

App.Trace("Bind to Tile Notification Exception : " + ex.Message);

throw ex;

}

notificationsBound = true;

}

1. UpdateStatus方法实现新描述客户端的连接状态的消息；ParseRAWPayload方法实现解析WPF服务应用程序发送的Raw通知。

Solution: PushNotificationsMango Project: WPPushNotification.TestClient

File: PushHandler.cs

private void UpdateStatus(string message)

{

Dispatcher.BeginInvoke(() => PushStatus.Message = message);

}

private void ParseRAWPayload(Stream e)

{

XDocument document;

using (var reader = new StreamReader(e))

{

string payload = reader.ReadToEnd().Replace('\0', ' ');

document = XDocument.Parse(payload);

}

XElement updateElement = document.Root;

string locationName = updateElement.Element("Location").Value;

LocationInformation locationInfo = Locations[locationName];

App.Trace("Got location: " + locationName);

string temperature = updateElement.Element("Temperature").Value;

locationInfo.Temperature = temperature;

App.Trace("Got temperature: " + temperature);

string weather = updateElement.Element("WeatherType").Value;

locationInfo.ImageName = weather;

App.Trace("Got weather type: " + weather);

}

1. 在**App.xaml**中我们添加了**Application.Resources**的Status属性，显示在应用程序的连接状态。

Solution: PushNotificationsMango Project: WPPushNotification.TestClient

File: App.xaml

<!--Application Resources-->

<Application.Resources>

<local:Status x:Key="PushStatus">

<local:Status.Message>Not connected</local:Status.Message>

</local:Status>

……

1. Application\_Launching方法实现了应用程序初始化启动时的位置信息，使用PushHandler类连接MSPN和WPF服务器。

Solution: PushNotificationsMango Project: WPPushNotification.TestClient

File: App.xaml.cs

// Code to execute when the application is launching (eg, from Start)

// This code will not execute when the application is reactivated

private void Application\_Launching(object sender, LaunchingEventArgs e)

{

TileRefreshNeeded = true;

InitializeLocations();

RefreshTilesPinState();

PushHandler = new PushHandler(Resources["PushStatus"] as Status, Locations, Dispatcher);

PushHandler.EstablishConnections();

}

InitializeLocations方法实现初始化位置信息，初始化了五个城市的位置信息：Redmond、Moscow、Paris、London和New York。

Solution: PushNotificationsMango Project: WPPushNotification.TestClient

File: App.xaml.cs

/// <summary>

/// Initializes the contents of the location dictionary.

/// </summary>

private void InitializeLocations()

{

List<LocationInformation> locationList = new List<LocationInformation>(new[] {

new LocationInformation { Name = "Redmond", TilePinned = false },

new LocationInformation { Name = "Moscow", TilePinned = false },

new LocationInformation { Name = "Paris", TilePinned = false },

new LocationInformation { Name = "London", TilePinned = false },

new LocationInformation { Name = "New York", TilePinned = false }

});

Locations = locationList.ToDictionary(l => l.Name);

}

1. Application\_Activated方法实现应用程序从逻辑删除情况下重新激活，还原位置信息，重新建立连接。如果应用程序的实例仍然保留，则检查连接是否存在，否则重建连接。最后，更新sub-tiles相关的位置信息。

Solution: PushNotificationsMango Project: WPPushNotification.TestClient

File: App.xaml.cs

// Code to execute when the application is activated (brought to foreground)

// This code will not execute when the application is first launched

private void Application\_Activated(object sender, ActivatedEventArgs e)

{

if (!e.IsApplicationInstancePreserved)

{

// The application was tombstoned, so restore its state

foreach (var keyValue in PhoneApplicationService.Current.State)

{

Locations[keyValue.Key] = keyValue.Value as LocationInformation;

}

// Reconnect to the MSPN

PushHandler = new PushHandler(Resources["PushStatus"] as Status, Locations, Dispatcher);

PushHandler.EstablishConnections();

}

else if (!PushHandler.ConnectionEstablished)

{

// Connection was not fully established before fast app switching occurred

PushHandler.EstablishConnections();

}

RefreshTilesPinState();

}

RefreshTilesPinState方法中使用的**ShellTile.ActiveTiles**返回所有应用程序的 sub-tiles和应用程序的主Tile。

Solution: PushNotificationsMango Project: WPPushNotification.TestClient

File: App.xaml.cs

/// <summary>

/// Sees which of the application's sub-tiles are pinned and updates the location information accordingly.

/// </summary>

private void RefreshTilesPinState()

{

Dictionary<string, LocationInformation> updateDictionary = Locations.Values.ToDictionary(li => li.Name);

foreach (ShellTile tile in ShellTile.ActiveTiles)

{

string[] querySplit = tile.NavigationUri.ToString().Split('=');

if (querySplit.Count() != 2)

{

continue;

}

string locationName = Uri.UnescapeDataString(querySplit[1]);

updateDictionary[locationName].TilePinned = true;

updateDictionary.Remove(locationName);

}

foreach (LocationInformation locationInformation in updateDictionary.Values)

{

locationInformation.TilePinned = false;

}

}

1. Application\_Deactivated方法实现在应用程序进入逻辑删除状态之前，保存所有位置信息。

Solution: PushNotificationsMango Project: WPPushNotification.TestClient

File: App.xaml.cs

// Code to execute when the application is deactivated (sent to background)

// This code will not execute when the application is closing

private void Application\_Deactivated(object sender, DeactivatedEventArgs e)

{

foreach (var keyValue in Locations)

{

PhoneApplicationService.Current.State[keyValue.Key] = keyValue.Value;

}

}

1. 更新客户端主页面

本节我们将修改MainPage.xaml，实现在MainPage显示任何订阅了sub-tiles通知的位置的天气信息。

1. **WPPushNotification.TestClient** 工程引用了**Microsoft.Phone.Controls.Toolkit**程序集。其添加方法是，在 **WPPushNotification.TestClient** 工程的References上点击右键，选择**[Add Reference…]**。

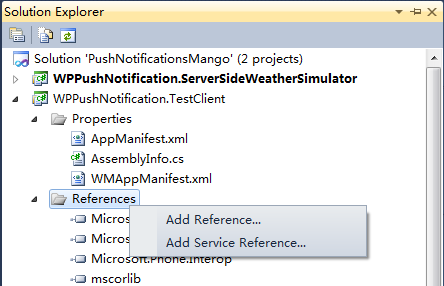


图7-7 添加引用

在弹出的窗体如下图，选择Browse选项卡，在**PushNotificationsMango\Assets\Lib**中选中**Microsoft.Phone.Controls.Toolkit.dll**。

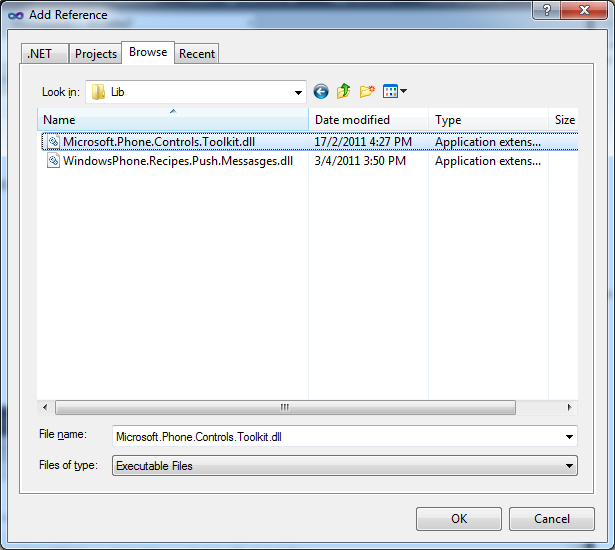


图7-8 Tookit引用

1. 打开**WPPushNotification.TestClient** 工程的**MainPage.xaml** ，如下面的代码所示，MaimPage.xaml引用了Microsoft.Phone.Controls.Toolkit 和WPPushNotification.TestClient.Converters 命名空间。

Solution: PushNotificationsMango Project: WPPushNotification.TestClient

File: MainPage.xaml

<phone:PhoneApplicationPage

x:Class="WPPushNotification.TestClient.MainPage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

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

xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"

xmlns:toolkit="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.Toolkit"

xmlns:conv="clr-namespace:WPPushNotification.TestClient.Converters"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"

FontFamily="{StaticResource PhoneFontFamilyNormal}"

FontSize="{StaticResource PhoneFontSizeNormal}"

Foreground="{StaticResource PhoneForegroundBrush}"

SupportedOrientations="Portrait" Orientation="Portrait"

shell:SystemTray.IsVisible="True"

DataContext="{Binding RelativeSource={RelativeSource Self}}">

1. 页面导航时MainPage从位置信息字典检索位置信息。

Solution: PushNotificationsMango Project: WPPushNotification.TestClient

File: MainPage.xaml.cs

/// <summary>

/// Contains information about the locations displayed by the application.

/// </summary>

public Dictionary<string, LocationInformation> Locations { get; set; }

// Constructor

public MainPage()

{

InitializeComponent();

}

protected override void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)

{

Locations = (App.Current as App).Locations;

base.OnNavigatedTo(e);

}

1. MakeTileUri方法将创建导航到CityPage的Uri，在Uri中位置信息是作为参数传递的。

Solution: PushNotificationsMango Project: WPPushNotification.TestClient

File: MainPage.xaml.cs

/// <summary>

/// Creates a Uri leading to the location specified by the location information to be bound to a tile.

/// </summary>

/// <param name="locationInformation">The location information for which to generate the Uri.</param>

/// <returns>Uri for the page displaying information about the provided location.</returns>

private static Uri MakeTileUri(LocationInformation locationInformation)

{

return new Uri(Uri.EscapeUriString(String.Format("/CityPage.xaml?location={0}",

locationInformation.Name)), UriKind.Relative);

}

1. 用户界面事件处理程序。UnpinItem\_Click和PinItem\_Click方法实现MainPage的位置信息关联的context菜单的事件处理；ListBox\_SelectionChanged方法实现导航到特定的位置信息的页面；ChangeMainTile\_Click方法实现更新主Tile通知。

Solution: PushNotificationsMango Project: WPPushNotification.TestClient

File: MainPage.xaml.cs

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

{

LocationInformation locationInformation = (sender as MenuItem).DataContext as LocationInformation;

ShellTile tile = ShellTile.ActiveTiles.FirstOrDefault(

t => t.NavigationUri.ToString().EndsWith(locationInformation.Name));

if (tile == null)

{

MessageBox.Show("Tile inconsistency detected. It is suggested that you restart the application.");

return;

}

try

{

tile.Delete();

locationInformation.TilePinned = false;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Error deleting tile", MessageBoxButton.OK);

return;

}

}

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

{

LocationInformation locationInformation = (sender as MenuItem).DataContext as LocationInformation;

Uri tileUri = MakeTileUri(locationInformation);

StandardTileData initialData = new StandardTileData()

{

BackgroundImage = new Uri("Images/Clear.png", UriKind.Relative),

Title = locationInformation.Name

};

((sender as MenuItem).Parent as ContextMenu).IsOpen = false;

try

{

ShellTile.Create(tileUri, initialData);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Error creating tile", MessageBoxButton.OK);

return;

}

}

private void ListBox\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (e.AddedItems.Count != 0)

{

(sender as ListBox).SelectedIndex = -1;

NavigationService.Navigate(MakeTileUri(e.AddedItems[0] as LocationInformation));

}

}

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

{

// Get the main tile (it will always be available, even if not pinned)

ShellTile mainTile = ShellTile.ActiveTiles.FirstOrDefault(t => t.NavigationUri.ToString() == "/");

StandardTileData newData = new StandardTileData()

{

BackgroundImage = new Uri(String.Format("Images/MainTile/{0}.png", (listMainTileImage.SelectedItem as ListPickerItem).Content), UriKind.Relative),

Title = txtMainTileTitle.Text

};

mainTile.Update(newData);

}

1. 调试应用程序

按F5运行客户端应用程序和WPF服务应用，或者点击Start Debugging按钮运行，如图7-9 Start Debugging。



图7-9 Start Debugging

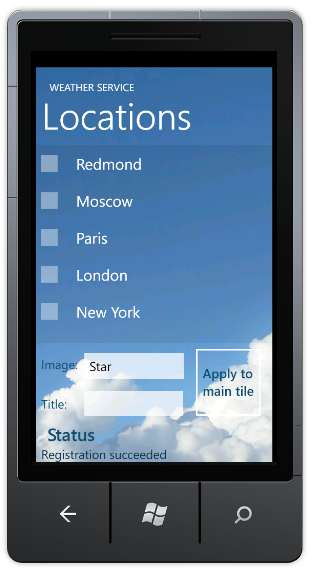


图7-10 主页面

按住城市名称，会弹出“Pin location”的菜单，确认后就可将选中的城市固定显示在启动屏幕中，如下图7-12 PinLocation所示。

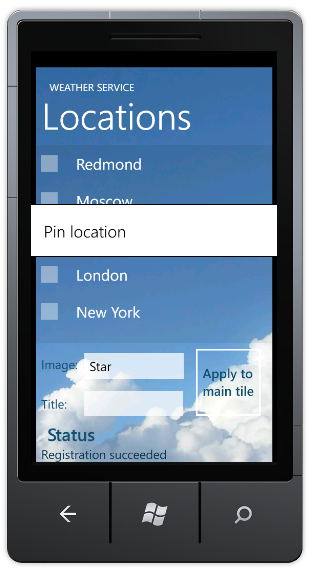
 

图7-12 PinLocation所示

按下后退键导航回应用程序，点击“Apply to main tile”按钮更新应用程序的主Tile。

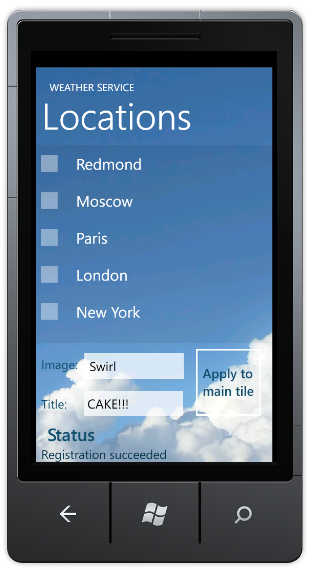
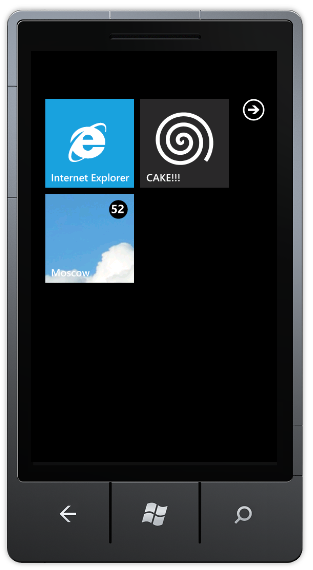
 

图7-13 更新应用程序的主Tile

### Deep toast通知

1. 指定城市的天气显示

将实现增强的tile和toast通知，并CityPage中实现显示特定城市的天气信息。

Solution: PushNotificationsMango Project: WPPushNotification.TestClient

File: CityPage.xaml

<Grid x:Name="LayoutRoot" Background="Transparent">

<Grid.Resources>

<conv:NameToImageConverter x:Key="NameToImageConverter"/>

</Grid.Resources>

<Grid.RowDefinitions>

<RowDefinition Height="120"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="150"/>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<Image Source="cloudbackgroundmobile.jpg" Grid.RowSpan="4" />

<Grid x:Name="TitleGrid" Grid.Row="0" VerticalAlignment="Top">

<TextBlock Text="WEATHER SERVICE" x:Name="textBlockPageTitle" Style="{StaticResource PhoneTextPageTitle1Style}" />

</Grid>

<Grid Grid.Row="1" x:Name="ContentPanel" Background="#10000000">

<TextBlock x:Name="textBlockListTitle" FontFamily="Segoe WP Light" FontSize="108" Text="{Binding Name}" Margin="20,10,0,0" />

<TextBlock x:Name="txtTemperature" FontFamily="Segoe WP" FontSize="160" Text="{Binding Temperature}" Margin="20,100,0,0" />

<Image x:Name="imgWeatherConditions" Width="128" Height="128" Stretch="None" HorizontalAlignment="Right" VerticalAlignment="Top" Margin="20,155,20,0" Source="{Binding ImageName, Converter={StaticResource NameToImageConverter}}" />

</Grid>

<StackPanel Grid.Row="3" x:Name="StatusStackPanel" Margin="20">

<TextBlock FontSize="34" FontFamily="Segoe WP Semibold" Foreground="#104f6f" Text="Status" Style="{StaticResource PhoneTextNormalStyle}" />

<TextBlock x:Name="txtStatus" DataContext="{StaticResource PushStatus}" Text="{Binding Message}" FontFamily="Segoe WP" FontSize="24" Foreground="#0a364c" Margin="0,0,0,0" Style="{StaticResource PhoneTextNormalStyle}" TextWrapping="Wrap" />

</StackPanel>

</Grid>

CityPage.xaml的布局显示城市的天气信息和背景图片，如图7-14 CityPage所示。

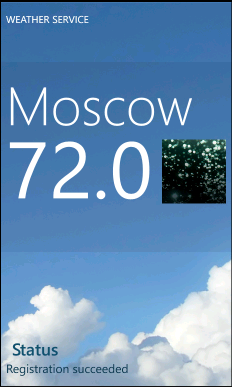


图7-14 CityPage

**NameToImageConverter**类实现天气状况的名称和图像的关联。

Solution: PushNotificationsMango Project: WPPushNotification.TestClient

File: Converters\NameToImageConverter.cs

namespace WPPushNotification.TestClient.Converters

{

public class NameToImageConverter : IValueConverter

{

#region IValueConverter Members

public object Convert(object value, Type targetType, object parameter, System.Globalization.CultureInfo culture)

{

return new BitmapImage(new Uri(String.Format("/Images/{0}.png", value), UriKind.Relative));

}

public object ConvertBack(object value, Type targetType, object parameter, System.Globalization.CultureInfo culture)

{

throw new NotImplementedException();

}

#endregion

}

}

CityPage类从导航的URI中获取location参数，将CityPage的数据源设置为location参数相关联的城市的天气信息。

Solution: PushNotificationsMango Project: WPPushNotification.TestClient

File: CityPage.xaml.cs

public partial class CityPage : PhoneApplicationPage

{

// Constructor

public CityPage()

{

InitializeComponent();

}

protected override void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)

{

DataContext = (App.Current as App).Locations[NavigationContext.QueryString["location"]];

base.OnNavigatedTo(e);

}

}

1. 更新服务

打开**MainWindow.xaml.csWPPushNotification.ServerSideWeatherSimulator**项目的**MainWindow.xaml.cs** 文件，**sendToast**方法实现服务器向应用程序选定的城市发送toast通知。

Solution: PushNotificationsMango Project: WPPushNotification.ServerSideWeatherSimulator

File: MainWindow.xaml.cs

private void sendToast()

{

string msg = txtToastMessage.Text;

txtToastMessage.Text = "";

List<Uri> subscribers = RegistrationService.GetSubscribers();

toastPushNotificationMessage.Title = String.Format("WEATHER ALERT ({0})", cmbLocation.SelectedValue);

toastPushNotificationMessage.SubTitle = msg;

toastPushNotificationMessage.TargetPage = MakeTileUri(cmbLocation.SelectedValue.ToString()).ToString();

subscribers.ForEach(uri => toastPushNotificationMessage.SendAsync(uri,

(result) => OnMessageSent(NotificationType.Toast, result),

(result) => { }));

}

sendTile方法将sub-tile发送到与通知相对应的城市。

Solution: PushNotificationsMango Project: WPPushNotification.ServerSideWeatherSimulator

File: MainWindow.xaml.cs

private void sendTile()

{

string weatherType = cmbWeather.SelectedValue as string;

int temperature = (int)(sld.Value + 0.5);

string location = cmbLocation.SelectedValue as string;

List<Uri> subscribers = RegistrationService.GetSubscribers();

tilePushNotificationMessage.BackgroundImageUri = new Uri("/Images/" + weatherType + ".png", UriKind.Relative);

tilePushNotificationMessage.Count = temperature;

tilePushNotificationMessage.Title = location;

tilePushNotificationMessage.SecondaryTile = MakeTileUri(location).ToString();

subscribers.ForEach(uri => tilePushNotificationMessage.SendAsync(uri,

(result) => OnMessageSent(NotificationType.Token, result),

(result) => { }));

}

sendRemoteTile方法实现发送背景图片资源位于远程服务器上的Tile通知。

Solution: PushNotificationsMango Project: WPPushNotification.ServerSideWeatherSimulator

File: MainWindow.xaml.cs

private void sendRemoteTile()

{

List<Uri> subscribers = RegistrationService.GetSubscribers();

tilePushNotificationMessage.BackgroundImageUri = new Uri(

"http://www.larvalabs.com/user\_images/screens\_thumbs/12555452181.jpg");

tilePushNotificationMessage.SecondaryTile = null;

tilePushNotificationMessage.Title = null;

tilePushNotificationMessage.Count = 0;

subscribers.ForEach(uri => tilePushNotificationMessage.SendAsync(uri,

(result) => OnMessageSent(NotificationType.Token, result),

(result) => { }));

}

服务器端的MakeTileUri方法与智能终端应用程序的构造URI的方法一致。

Solution: PushNotificationsMango Project: WPPushNotification.ServerSideWeatherSimulator

File: MainWindow.xaml.cs

/// <summary>

/// Creates a Uri leading to the location specified by the location information to be bound to a tile.

/// </summary>

/// <param name="locationName">The name of the location for which the Uri is constructed.</param>

/// <returns>Uri for the page displaying information about the provided location.</returns>

private static Uri MakeTileUri(string locationName)

{

return new Uri(Uri.EscapeUriString(String.Format("/CityPage.xaml?location={0}",

locationName)), UriKind.Relative);

}

1. 调试应用程序

按F5运行客户端应用程序和WPF服务应用，或者点击Start Debugging按钮运行，如图7-15 Start Debugging。



图7-15 Start Debugging

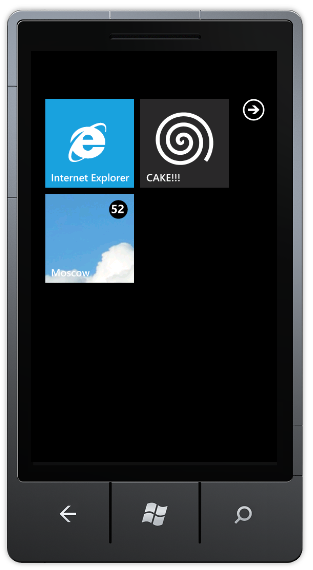


图7-16 start screen

在WPF服务器端应用程序发送toast通知，并在智能终端客户端点击固定显示在启动屏幕（start screen）的天气预报应用程序图标。

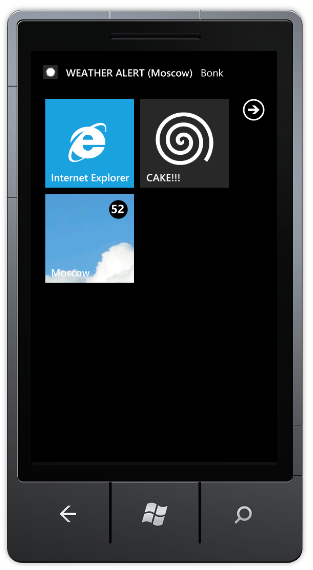
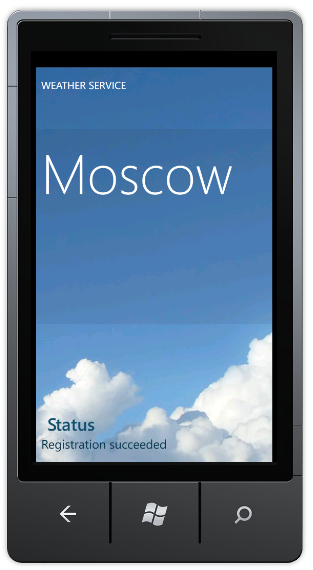
 

图7-17 toast通知

发送raw通知，则在CityPage页面显示raw通知的相关数据信息。

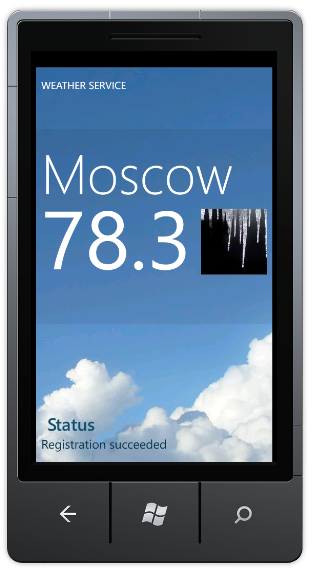


图7-18 指定城市的天气信息

## 动手实践——深度分析推送通知实现架构（Windows Phone OS 7.0）

Windows Phone应用程序的推送通知的实现方式中，程序员几乎不需要编写代码就可以实现在Windows Phone的三种推送通知响应，原因是系统本身已经替我们做好了。如果应用程序使用推送通知功能的话，需要开发者关注的主要有两个方面，第一是启用和关闭应用程序推送通知的设定，因为这是MarketPlace要求应用程序必须具备的功能；第二是Web Service的设计和代码实现，因为实现推送通知消息内容的逻辑都是在Web Service端完成的，然后通知MPNS将消息推送至Windows Phone应用程序。Web Service既可以是云端的Cloud Application，也可以是其他的Web应用程序，只要能和MPNS通讯即可。

本节中，我们参考微软官方博客--The Windows Blog上的文章Windows Push Notification Server Side Helper Library，深度解析推送通知实现架构中的需要开发者重点关注的两个方面。

### Windows Phone推送通知类型（Windows Phone OS 7.0）

Windows Phone中存在三种默认通知类型：Tile、Push 和 Toast 通知。

下图显示了Tile和Toast通知之间的差异：

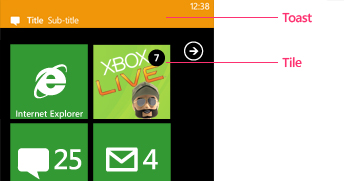


图7-19 Tile和Toast的区别

1. [Tile](javascript:void(0))通知

每个应用程序可设置Tile—应用程序内容的可视化、 动态的表示形式。当应用程序被固定显示在启动屏幕(Start Screen)时，我们就可以看到Tile的信息。Tile可以修改的三个元素包括：计数(Count)、标题(Title)和背景图像(Background)。

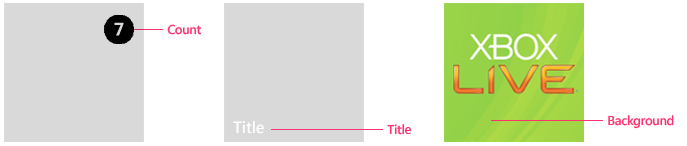


图7-20 Tile通知

1. 背景图像

您可以使用本地资源或远程资源的背景图像。如果要使用本地资源，则它必须是安装包XAP的一部分，XAP即为Windows Phone的安装包。例如，它不可能下载图像，把它放入独立存储，然后使用它作为本地资源的背景图像的拼贴。为了获得最佳性能，请考虑使用本地资源。

背景图像将永远不会恢复到以前的版本后，它已成功更新，除非使用推式通知再次发送了以前的背景图像。

1. 标题

标题必须适应单行文本，字符串不应过长大于实际Tile。如果不设置标题，则默认显示现有标题。

1. 计数

计数为整数值 1 到 99。如果通知中未设置计数值，或设置超出范围整数值，则当前计数值将继续显示。例如，如果当前通知中没有设置计数的值，那么计数的显示会保持不变：不显示计数或者显示上一次通知的计数。要清除计数显示就必须设置计数的值为0。

1. [[http://i.msdn.microsoft.com/Hash/030c41d9079671d09a62d8e2c1db6973.gif](javascript:void(0))Toast通知](javascript:void(0))

Toast通知是Windows Phone系统通知，且不破坏用户的工作流，十秒钟后自动消失。Toast通知显示在屏幕的顶部。

Toast通知的两个文本元素：标题和副标题。标题为粗体字显示的字符串，副标题为非粗体字显示的字符串。

|  |
| --- |
| Important note**重要说明：** |
| 您必须要求用户授权方可接收Toast通知，且在应用程序中必须具有允许用户禁用的Toast通知的功能。 |

1. [Raw通知](javascript:void(0))

Raw通知的格式可以任意设定。如果当前没有运行您的应用程序，Raw通知将被微软推通知服务丢弃，不会传递到Windows Phone设备。Raw通知的有效载荷的最大为 1 KB。

### 推送通知的工作流（Windows Phone OS 7.0）

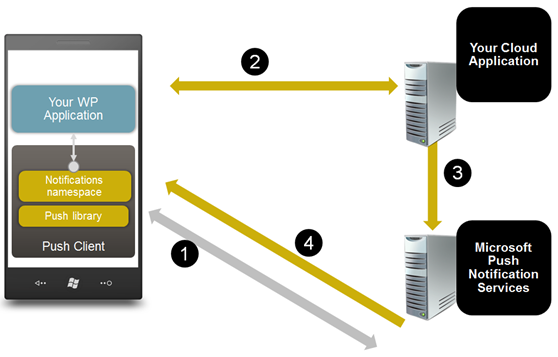


图7-21 推送通知工作流

1. Window Phone客户端应用程序请求与微软推送通知服务(Microsoft Push Notification Services)建立通道连接，微软推送通知服务(Microsoft Push Notification Services)使用通道**URI**响应。
2. 客Window Phone客户端应用程序向监视服务(Web Service或者Cloud Application)发送包含推送通知服务通道**URI**以及负载的消息。
3. 当监视服务检测到信息更改时（如航班取消、航班延期或天气警报），它会向微软推送通知服务(Microsoft Push Notification Services)发送消息。
4. 微软推送通知服务(Microsoft Push Notification Services)将消息中继到Windows Phone设备，由Window Phone客户端应用程序处理收到的消息。

### 推送通知的消息类

推送通知消息基础类：**PushNotificationMessage**类，以及三个子类：**RawPushNotificationMessage**、 **TilePushNotificationMessage**和**ToastPushNotificationMessage** 。如图 *Push Messages Class Diagram*。

* **RawPushNotificationMessage** – 当Windows Phone应用程序运行时，可以接收到来自Web Service的Raw通知消息。
* **TilePushNotificationMessage** –当Windows Phone应用程序被固定显示在启动页面，Windows Phone将呈现Tile通知消息的内容。
* **ToastPushNotificationMessage** –发送Toast“警告”消息至Windows Phone。



图7-22 Push Messages Class Diagram

### 发送Tile通知

下面的代码片段演示了如何使用Windows Phone的推送通知类库以同步和异步的方式发送Tile通知。

C#

// Prepare a tile push notification message.

var tile = new TilePushNotificationMessage

{

BackgroundImageUri = tileImageUri, // Remote or phone-local tile image uri.

Count = tileCount, // Counter between 1 to 99 should be displayed on the tile.

Title = “Tile Title” // Title to be displayed on the tile.

};

// Send the message synchronously.

try

{

var sendResult = tile.Send(phoneChannelUri);

// Check the send result.

}

catch (Exception ex)  
{

// Log the error.  
}

// Send the message asynchronously.

tile.SendAsync(

phoneChannelUri,

result => {/\* Check the send result \*/},

exception => {/\* Log the error \*/});

从上面的代码可以看出，发送Tile通知到Windows Phone应用多么简单，仅仅是创建一个新的TilePushNotificationMessage，并设置相关的属性，然后调用同步Send和异步SendAsync的方法即可。

### 发送Toast通知

下面的代码片段演示了如何使用Windows Phone的推送通知类库以同步和异步的方式发送Toast通知。

C#

// Prepare a toast push notification message.

var toast = new ToastPushNotificationMessage

{

Title = “Title”, // Title to be displayed as the toast header.

Subtitle = “Sub Title” // Message to be displayed next to the toast header.

};

// Send the message synchronously.

try

{

var sendResult = toast.Send(phoneChannelUri);

// Check the send result.

}

catch (Exception ex)

{

// Log the error.

}

// Send the message asynchronously.

toast. SendAsync (

phoneChannelUri,

result => { /\* Check the send result \*/ },

exception => { /\* Log the error \*/ });



### 发送Raw通知

下面的代码片段演示了如何使用Windows Phone的推送通知类库以同步和异步的方式发送Raw通知。

C#

// Prepare a raw push notification message.

byte[] rawData = {};

var raw = new RawPushNotificationMessage

{

RawData = rawData, // Raw data to be sent with the message.

};

// Send the message synchronously.

try

{

var sendResult = raw.Send(phoneChannelUri);

// Check the send result.

}

catch (Exception ex)  
{

// Log the error.  
}

// Send the message asynchronously.

raw. SendAsync (

phoneChannelUri,

result => { /\* Check the send result \*/ },

exception => { /\* Log the error \*/ });

### 客户端设定启动推送通知

我们使用简单的Silverlight应用说明如何将推送通知消息发送到Windows Phone智能手机。

1. 注册推送通知服务

为了接收推送通知消息，Windows Phone应用程序需要向微软推送通知服务MPNS发送注册请求，MPNS返回的Windows Phone智能手机的URI（统一资源标识符）。

在我们的示例中，推送通知PN功能封装在**PushContext**类。强烈建议您每次启动应用程序时将PN通道URI更新到Web Service，确保您的Web Service中保存的是智能手机最新的URI。

1. 推送通知设置页面

每个Windows Phone应用，如果使用推送通知都必须允许用户设置推送通知功能的开启和关闭。 即程序开启推送通知PN通道时需要得到用户的许可，且用户也可以选择关闭推送通知PN通道。下面的画面(Project : WindowsPhone.Recipes.Push.Client File : Views/PushSettingControl.XAML)显示了示例应用程序的推送通知设定页面。

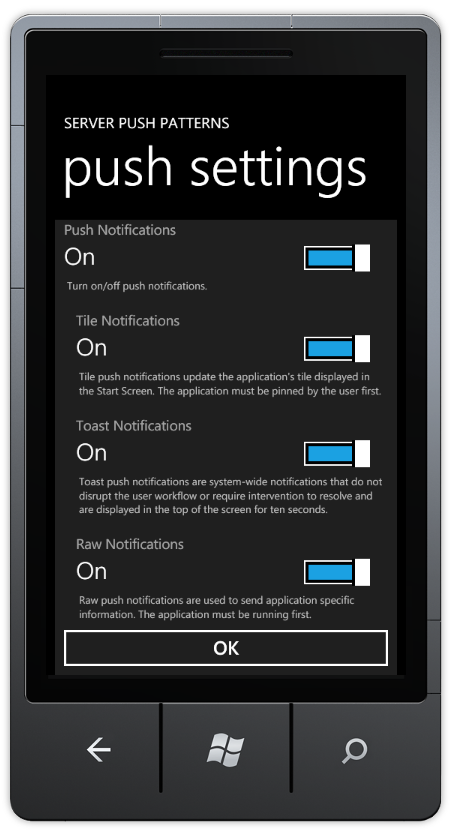


图7-23 push setting

Project : WindowsPhone.Recipes.Push.Client File : Views/PushSettingControl.XAML

<UserControl x:Class="WindowsPhone.Recipes.Push.Client.Controls.PushSettingsControl"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:tk="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.Toolkit"

xmlns:converters="clr-namespace:WindowsPhone.Recipes.Push.Client.Converters"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d"

FontFamily="{StaticResource PhoneFontFamilyNormal}"

FontSize="{StaticResource PhoneFontSizeNormal}"

Foreground="{StaticResource PhoneForegroundBrush}"

d:DesignHeight="640" d:DesignWidth="480">

<UserControl.Resources>

<converters:BoolBrushConverter x:Key="BoolBrushConverter" />

<Style x:Key="DescTextStyle" TargetType="TextBlock">

<Setter Property="FontSize" Value="14" />

<Setter Property="Foreground" Value="Silver" />

<Setter Property="TextWrapping" Value="Wrap" />

<Setter Property="Margin" Value="16,-38,16,24" />

</Style>

</UserControl.Resources>

<Grid x:Name="LayoutRoot" Background="{StaticResource PhoneChromeBrush}">

<StackPanel>

<StackPanel>

<tk:ToggleSwitch Header="Push Notifications"

IsChecked="{Binding IsPushEnabled, Mode=TwoWay}" />

<TextBlock Style="{StaticResource DescTextStyle}"

Text="Turn on/off push notifications." />

</StackPanel>

<Grid>

<StackPanel Margin="16,0,0,0">

<tk:ToggleSwitch Header="Tile Notifications"

IsChecked="{Binding IsTileEnabled, Mode=TwoWay}" />

<TextBlock Style="{StaticResource DescTextStyle}"

Text="Tile push notifications update the application's tile displayed in the Start Screen. The application must be pinned by the user first." />

<tk:ToggleSwitch Header="Toast Notifications"

IsChecked="{Binding IsToastEnabled, Mode=TwoWay}" />

<TextBlock Style="{StaticResource DescTextStyle}"

Text="Toast push notifications are system-wide notifications that do not disrupt the user workflow or require intervention to resolve and are displayed in the top of the screen for ten seconds." />

<tk:ToggleSwitch Header="Raw Notifications"

IsChecked="{Binding IsRawEnabled, Mode=TwoWay}" />

<TextBlock Style="{StaticResource DescTextStyle}"

Text="Raw push notifications are used to send application specific information. The application must be running first." />

</StackPanel>

<Border Background="{Binding IsPushEnabled, Converter={StaticResource BoolBrushConverter}}" />

</Grid>

</StackPanel>

</Grid>

</UserControl>

当用户登录时，注册PN推送通知的通道。

Project: WindowsPhone.Recipes.Push.Client File: Views/ UserLoginView.xaml.cs

private void InternalLogin()

{

login.Visibility = Visibility.Collapsed;

progress.Visibility = Visibility.Visible;

var pushContext = PushContext.Current;

pushContext.Connect(c => RegisterClient(c.ChannelUri));

}

private void RegisterClient(Uri channelUri)

{

// Register the URI with 3rd party web service.

try

{

var pushService = new PushServiceClient();

pushService.RegisterCompleted += (s, e) =>

{

pushService.CloseAsync();

Completed(e.Error);

};

pushService.RegisterAsync(UserName, channelUri);

}

catch (Exception ex)

{

Completed(ex);

}

}

向Web Service提交订阅，创建PN通道

Project: WindowsPhone.Recipes.Push.Client File: PushContext.cs

public void Connect(Action<HttpNotificationChannel> prepared)

{

if (IsConnected)

{

prepared(NotificationChannel);

return;

}

try

{

// First, try to pick up an existing channel.

NotificationChannel = HttpNotificationChannel.Find(ChannelName);

if (NotificationChannel == null)

{

// Create new channel and subscribe events.

CreateChannel(prepared);

}

else

{

// Channel exists, no need to create a new one.

SubscribeToNotificationEvents();

PrepareChannel(prepared);

}

IsConnected = true;

}

catch (Exception ex)

{

OnError(ex);

}

}

public void Disconnect()

{

if (!IsConnected)

{

return;

}

try

{

if (NotificationChannel != null)

{

UnbindFromTileNotifications();

UnbindFromToastNotifications();

NotificationChannel.Close();

}

}

catch (Exception ex)

{

OnError(ex);

}

finally

{

NotificationChannel = null;

IsConnected = false;

}

}

创建PN通道的具体函数。

Project: WindowsPhone.Recipes.Push.Client File: PushContext.cs

/// <summary>

/// Create channel, subscribe to channel events and open the channel.

/// </summary>

private void CreateChannel(Action<HttpNotificationChannel> prepared)

{

// Create a new channel.

NotificationChannel = new HttpNotificationChannel(ChannelName, ServiceName);

// Register to UriUpdated event. This occurs when channel successfully opens.

NotificationChannel.ChannelUriUpdated += (s, e) => Dispatcher.BeginInvoke(() => PrepareChannel(prepared));

SubscribeToNotificationEvents();

// Trying to Open the channel.

NotificationChannel.Open();

}

绑定和解除绑定Raw、Tile和Toast通知消息的函数。

Project: WindowsPhone.Recipes.Push.Client File: PushContext.cs

private void BindToTileNotifications()

{

try

{

if (NotificationChannel != null && !NotificationChannel.IsShellTileBound)

{

var listOfAllowedDomains = new Collection<Uri>(AllowedDomains);

NotificationChannel.BindToShellTile(listOfAllowedDomains);

}

}

catch (Exception ex)

{

OnError(ex);

}

}

private void BindToToastNotifications()

{

try

{

if (NotificationChannel != null && !NotificationChannel.IsShellToastBound)

{

NotificationChannel.BindToShellToast();

}

}

catch (Exception ex)

{

OnError(ex);

}

}

private void UnbindFromTileNotifications()

{

try

{

if (NotificationChannel.IsShellTileBound)

{

NotificationChannel.UnbindToShellTile();

}

}

catch (Exception ex)

{

OnError(ex);

}

}

private void UnbindFromToastNotifications()

{

try

{

if (NotificationChannel.IsShellToastBound)

{

NotificationChannel.UnbindToShellToast();

}

}

catch (Exception ex)

{

OnError(ex);

}

}

### Web Service设计推送通知功能

为了说明如何在Web Service设计推送通知服务功能，我们创建WPF应用程序Push Notifications Server，即Web Service，模拟第三方服务器。 在WPF应用程序创建一个WCF服务，让Windows Phone应用订阅通知服务并与MPNS通讯。

WPF应用程序**Push Notifications Server**有五个选项卡（One-time、 Ask to Pin、 Custom Tile、Counter和Tile Scheduled），One-Time实现一次性推送三种类型的通知消息；Counter演示计算器重置；Ask to Pin实现询问用户是否将应用程序固定显示在启动页面；Custom Tile实现定制Tile通知消息Tile Scheduled实现设定Tile更新计划表。下面的章节将详细讲述这五个选项卡的功能。

1. One-Time
2. 说明

One-Time选项卡是向注册用户提供最简单的推送模式，即一次性将三种类型的推送通知：Raw、Tile和Toast通知推送给Windows Phone。One-Time选项卡显示了三种类型通知的可设置的属性，以及MPNS的返回值类型。

1. 操作
2. 运行 WindowsPhone.Recipes.Push.Server 和WindowsPhone.Recipes.Push.Client projects (设置 WindowsPhone.Recipes.Push.Server为默认启动程序)。
3. 在Windows Phone模拟器中以任意用户名登录。

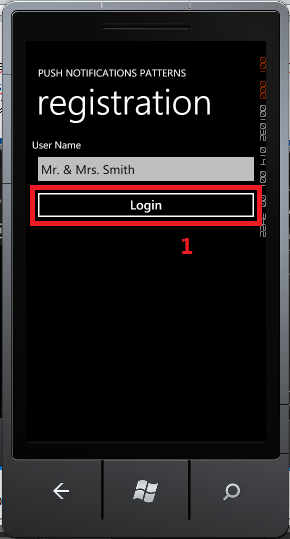
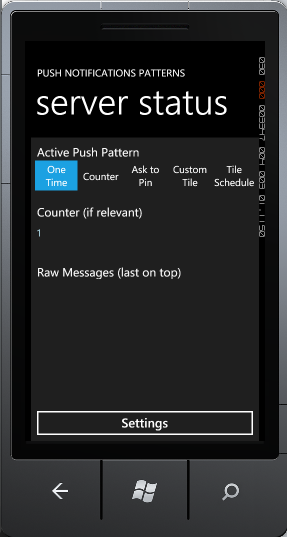
 

图7-24 登陆

1. 在服务器端，选择“One Time”选项卡。

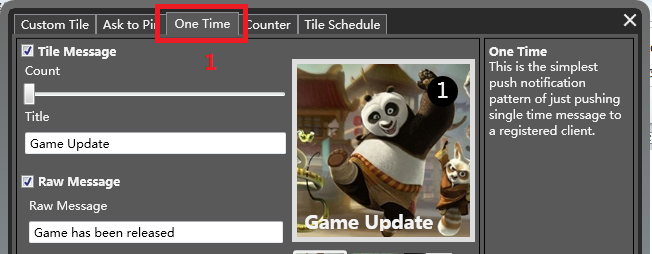


图7-25 One Time选项卡

1. 设置Raw、Tile和Toast消息内容，然后选择“Send”发送。

**注意：**

只有Windows Phone应用程序正在运行，Windows Phone应用程序才会接收到Raw通知；当Windows Phone应用程序不在运行状态时，则会显示Toast通知；当Windows Phone应用程序被固定在启动页面上时，就会显示接收到的Tile消息。

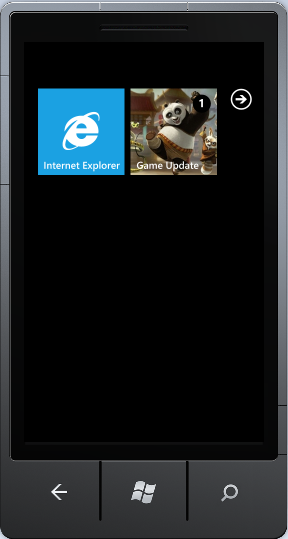
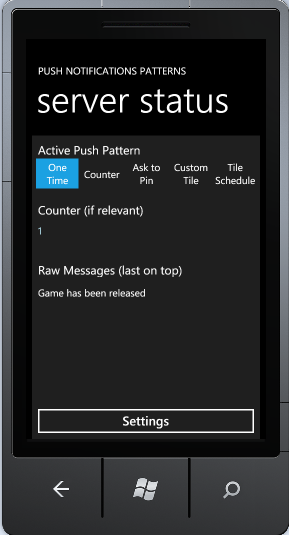
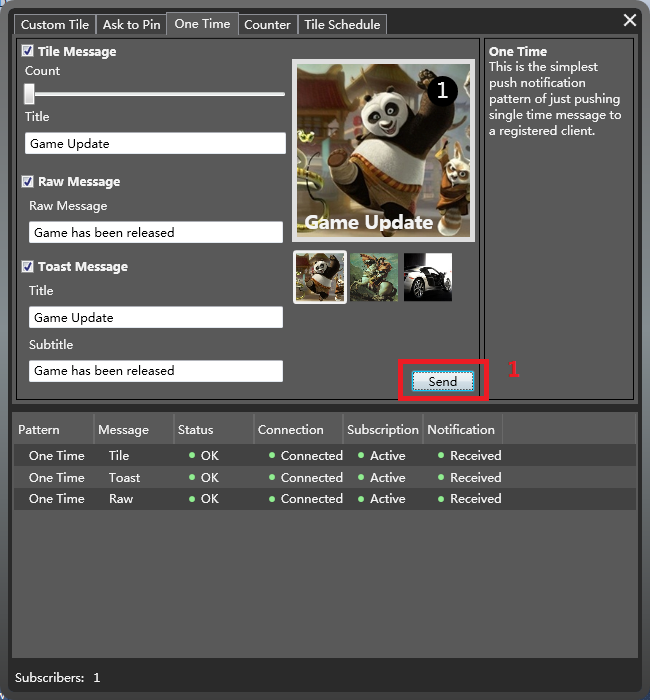


图7-26 发送通知

1. 代码

在我们的例子，WPF应用程序**Push Notifications Server**即Web Service设定各种通知的内容，然后以异步的方式将消息发送到相关的客户端。

Project: WindowsPhone.Recipes.Push.Server

File: ViewModels/Patterns/OneTimePushPatternViewModel.cs

/// <summary>

/// Depends on what message was selected, send all subscribers zero or all three push message types (Tile, Toast, Raw).

/// </summary>

protected override void OnSend()

{

var messages = new List<PushNotificationMessage>();

if (IsTileEnabled)

{

// Prepare a tile push notification message.

messages.Add(new TilePushNotificationMessage(MessageSendPriority.High)

{

BackgroundImageUri = BackgroundImageUri,

Count = Count,

Title = Title

});

}

if (IsToastEnabled)

{

// Prepare a toast push notification message.

messages.Add(new ToastPushNotificationMessage(MessageSendPriority.High)

{

Title = ToastTitle,

SubTitle = ToastSubTitle

});

}

if (IsRawEnabled)

{

// Prepare a raw push notification message.

messages.Add(new RawPushNotificationMessage(MessageSendPriority.High)

{

RawData = Encoding.ASCII.GetBytes(RawMessage)

});

}

foreach (var subscriber in PushService.Subscribers)

{

messages.ForEach(m => m.SendAsync(subscriber.ChannelUri, Log, Log));

}

}

1. 计数器重置
2. 说明

Windows Phone启动页面呈现应用程序的Tile信息，如标题、图片和计数器。 如电子邮件程序显示未读邮件的数量。在本例的Counter选项卡中，每次Tile通知消息发送，计数器加1。 下次用户登录到服务器端的应用程序时，计数器复位。

1. 代码

在本例中，OnSend方法创建一个Raw通知并发送到所有订阅的Windows Phone智能手机。 发送完成后，调用OnRawSent方法。

Project: WindowsPhone.Recipes.Push.Server

File: ViewModels/Patterns/CounterPushPatternViewModel.cs

/// <summary>

/// Send raw message to all subscribers. In case that the phone-application

/// is not running, send tile update and increase tile counter.

/// </summary>

protected override void OnSend()

{

// Notify phone for having waiting messages.

var rawMsg = new RawPushNotificationMessage(MessageSendPriority.High)

{

RawData = Encoding.ASCII.GetBytes(RawMessage)

};

foreach (var subscriber in PushService.Subscribers)

{

rawMsg.SendAsync(

subscriber.ChannelUri,

result =>

{

Log(result);

OnRawSent(subscriber.UserName, result);

},

Log);

}

}

一个Raw消息被发送后，运行回调函数**OnRawSent。** 获取MPNS的检查该设备是否连接的返回值。 如果手机没有连接，发送一个Raw消息是没有意义。 如果设备已连接，则发送一个Tile通知提示用户，Tile通知的内容就是计数器加1。

Project: WindowsPhone.Recipes.Push.Server

File: ViewModels/Patterns/ CounterPushPatternViewModel.cs

private void OnRawSent(string userName, MessageSendResult result)

{

// In case that the device is disconnected, no need to send a tile message.

if (result.DeviceConnectionStatus == DeviceConnectionStatus.TempDisconnected)

{

return;

}

// Checking these three flags we can know what's the state of both the device and apllication.

bool isApplicationRunning =

result.SubscriptionStatus == SubscriptionStatus.Active &&

result.NotificationStatus == NotificationStatus.Received &&

result.DeviceConnectionStatus == DeviceConnectionStatus.Connected;

// In case that the application is not running, send a tile update with counter increase.

if (!isApplicationRunning)

{

var tileMsg = new TilePushNotificationMessage(MessageSendPriority.High)

{

Count = IncreaseCounter(userName),

BackgroundImageUri = BackgroundImageUri,

Title = Title

};

tileMsg.SendAsync(result.ChannelUri, Log, Log);

}

}

Windows Phone应用程序重新登陆后，WPF应用程序**Push Notifications Server**创建一个计数器清零的Tile通知Windows Phone应用程序的计数器复位。

Project: WindowsPhone.Recipes.Push.Server

File: ViewModels/Patterns/ CounterPushPatternViewModel.cs

/// <summary>

/// On subscription change, reset the subscriber tile counter if exist.

/// </summary>

protected override void OnSubscribed(SubscriptionEventArgs e)

{

// Create a tile message to reset tile count.

var tileMsg = new TilePushNotificationMessage(MessageSendPriority.High)

{

Count = 0,

BackgroundImageUri = BackgroundImageUri,

Title = Title

};

tileMsg.SendAsync(e.Subscription.ChannelUri, Log, Log);

ResetCounter(e.Subscription.UserName);

}

1. Ask to Pin
2. 说明

假设Windows Phone应用程序没有被固定显示在启动页面上，Tile通知将会不显示。在这种情况下，Web Service通常会询问用户是否将应用程序在Windows Phone启动页面上呈现，如图7-27 询问用户。为此，需要开发Web Service和Windows Phone客户端的代码。

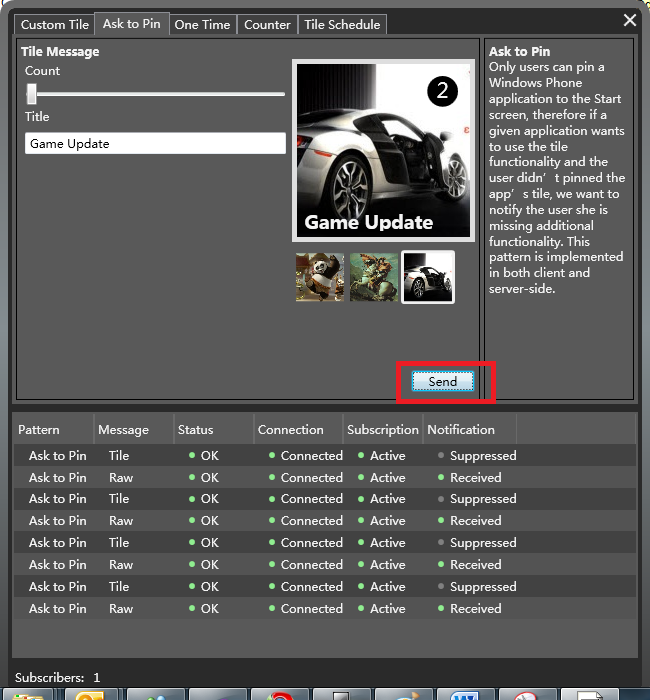
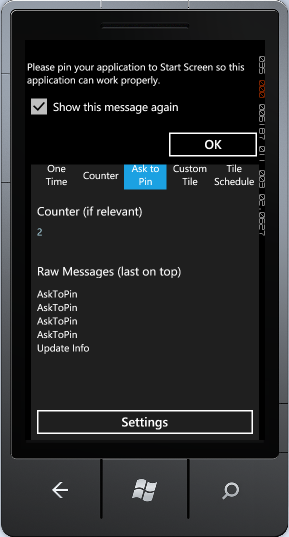
 

图7-27 询问用户

1. 代码

为了检查Windows Phone应用程序是否固定显示在启动页面，OnSubscribed方法发送一个Tile通知消息给Windows Phone客户端应用程序（基于Windows Phone的URI和用户姓名）。

Project: WindowsPhone.Recipes.Push.Server

File: ViewModels/Patterns/ AskToPinPushPatternViewModel.cs

/// <summary>

/// Once an application is activated again (the client side phone application

/// has subscription logic on startup), try to update the tile again.

/// In case that the application is not pinned, send raw notification message

/// to the client, asking to pin the application. This raw notification message

/// has to be well-known and handled by the client side phone application.

/// In our case the raw message is AskToPin.

/// </summary>

protected override void OnSubscribed(SubscriptionEventArgs args)

{

// Asynchronously try to send Tile message to the relevant subscriber

// with data already sent before so the tile won't change.

var tileMsg = GetOrCreateMessage(args.Subscription.UserName, false);

tileMsg.SendAsync(

args.Subscription.ChannelUri,

result =>

{

Log(result);

OnMessageSent(args.Subscription.UserName, result);

},

Log);

}

上面的代码中，Tile通知发送的是异步消息，回调OnMessageSent方法。WPF应用程序**Push Notifications Server**检查MPNS的返回消息，确定Windows Phone应用程序是否固定在启动页面上。如果已经固定呈现在启动页面上，则不执行任何操作。如果否，那么就发送一个Raw通知消息，提示用户将Windows Phone应用程序显示在启动页面上。

Project: WindowsPhone.Recipes.Push.Server

File: ViewModels/Patterns/ AskToPinPushPatternViewModel.cs

/// <summary>

/// Once tile update sent, check if handled by the phone.

/// In case that the application is not pinned, ask to pin.

/// </summary>

private void OnMessageSent(string userName, MessageSendResult result)

{

if (!CheckIfPinned(result))

{

AskUserToPin(result.ChannelUri);

}

}

/// <summary>

/// Just in case that the application is running, send a raw message, asking

/// the user to pin the application. This raw message has to be handled in client side.

/// </summary>

private void AskUserToPin(Uri uri)

{

new RawPushNotificationMessage(MessageSendPriority.High)

{

RawData = Encoding.ASCII.GetBytes(RawMessage)

}.SendAsync(uri, Log, Log);

}

通过MPNS返回值的三个标志DeviceConnectionStatus、SubscriptionStatus和NotificationStatus来确定应用程序是否被固定显示在启动页面。

Project: WindowsPhone.Recipes.Push.Server

File: ViewModels/Patterns/ AskToPinPushPatternViewModel.cs

private bool CheckIfPinned(MessageSendResult result)

{

// We known if the application is pinned by checking the following send result flags:

return result.DeviceConnectionStatus == DeviceConnectionStatus.Connected &&

result.SubscriptionStatus == SubscriptionStatus.Active &&

result.NotificationStatus == NotificationStatus.Received;

}

1. 定制Tile通知消息
2. 说明

在启动页面上的Tile通知的图像是动态的，可以是应用程序内置的图片，也可以是网络上的图片，只要图片的URI地址是有效的即可。

1. 限制

当Tile指向图像的URI是远程服务器时请注意下列限制：

* URI必须能够访问到手机
* 图像的大小必须小于80KB
* 下载的时间不能超过60秒

1. 代码

本节中，发送Tile通知消息的OnSend方法，设定Tile通知消息的URI信息为[http://localhost:8000/ImageService/GetTileImage?uri=channel\_uri](http://translate.google.com/translate?hl=zh-CN&sl=en&tl=zh-CN&prev=_t&u=http://localhost:8000/ImageService/GetTileImage%3Furi%3Dchannel_uri) ，而不是一个We bService的URL图像位置。

因为Tile图像URI必须指向一个远程地址的服务器，如Web上的图像资源。在本例子中为了演示方便我们偷梁换柱，通过WCF ImageService类将图像的URL发送给REST服务。 这个REST服务就是GetTileImage REST服务，它返回图片的Stream对象。

Project: WindowsPhone.Recipes.Push.Server

File: ViewModels/Patterns/ CustomTileImagePushPatternViewModel.cs

protected override void OnSend()

{

// Starts by sending a tile notification to all relvant subscribers.

// This tile notification updates the tile with custom image.

var tileMsg = new TilePushNotificationMessage(MessageSendPriority.High)

{

Count = Count,

Title = Title

};

foreach (var subscriber in PushService.Subscribers)

{

// Set the tile background image uri with the address of the ImageService.GetTileImage,

// REST service, using current subscriber channel uri as a parameter to bo sent to the service.

tileMsg.BackgroundImageUri = new Uri(string.Format(ImageService.GetTileImageService, string.Empty));

tileMsg.SendAsync(subscriber.ChannelUri, Log, Log);

}

}

发送Tile通知消息后，Windows Phone返回URL激活REST服务：ImageService.GetTileImage。 这个方法会触发一个事件，要求WPF应用程序提供的图像流。 在实际的应用程序中，您可以使用图像处理库创建定制的图像而不需要用到本例中关于UI的技术。

Project: WindowsPhone.Recipes.Push.Server

File: Services / ImageService.cs

/// <summary>

/// Get a generated custom tile image stream for the given uri.

/// </summary>

/// <param name="parameter">The tile image request parameter.</param>

/// <returns>A stream of the custom tile image generated.</returns>

public Stream GetTileImage(string parameter)

{

if (ImageRequest != null)

{

var args = new ImageRequestEventArgs(parameter);

ImageRequest(this, args);

// Seek the stream back to the begining just in case.

args.ImageStream.Seek(0, SeekOrigin.Begin);

return args.ImageStream;

}

return Stream.Null;

}

1. Tile更新计划表
2. 说明

Web Service可以通过微软推送服务发送Tile通知时，Windows Phone也使用ShellTileSchedule类定期检查Tile更新。 Windows Phone应用程序可以通过唯一标识URI自动发送请求，定期获取Tile的更新。

1. 代码

使用**ShellTileSchedule**类设定Tile通知的定时更新。目前Windows Phone智能手机支持的最频繁更新周期是每小时更新。本例中，**ShellTileSchedule**类设定的设定方法为，点击“设定计划表”设定更新计划。

Project: WindowsPhone.Recipes.Push.Client

File: Views/InboxView.xaml.cs

private ShellTileSchedule \_tileSchedule;

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

{

\_tileSchedule = new ShellTileSchedule();

\_tileSchedule.Recurrence = UpdateRecurrence.Interval;

\_tileSchedule.StartTime = DateTime.Now;

\_tileSchedule.Interval = UpdateInterval.EveryHour;

\_tileSchedule.RemoteImageUri = new Uri(string.Format(GetTileImageService, TileScheduleParameter));

\_tileSchedule.Start();

}

点击“测试URI”按钮会导致WPF应用程序**Push Notifications Server**发送Tile通知消息。运行此功能前需要将Windows Phone应用程序Push Patterns固定显示在启动页面。

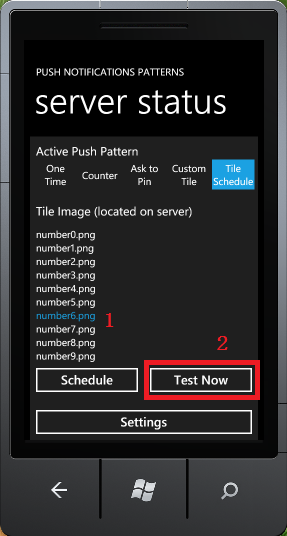
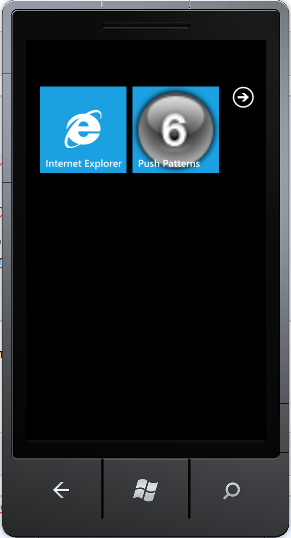
  

图7-28 发送Tile通知

Project: WindowsPhone.Recipes.Push.Client

File: Views/InboxView.xaml.cs

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

{

try

{

var pushService = new PushServiceClient();

pushService.UpdateTileCompleted += (s1, e1) =>

{

try

{

pushService.CloseAsync();

}

catch (Exception ex)

{

ex.Show();

}

};

pushService.UpdateTileAsync(PushContext.Current.NotificationChannel.ChannelUri, SelectedServerImage);

}

catch (Exception ex)

{

ex.Show();

}

}