马宁的Windows Phone 7.1初体验（二）——Push Notification

转载

weixin_33774308 于 2017-10-16 19:10:00 发布 80 收藏
文章标签： 操作系统 移动开发 shell
原文链接：https://yq.aliyun.com/articles/398171

Push Notification并不是Windows Phone 7.1的新功能，但是之前的文章里对这部分都缺少详细的分析，所以姑且就把Push Notification放到这部分里吧。

很多iOS开发者都将WP7里的Push Notification说成抄袭iOS的产物，孰不知，微软才是Push Notification技术的先行者，Windows Mobile时代的Push Mail技术可以说是独步天下，连Symbian也要授权使用相关的技术。

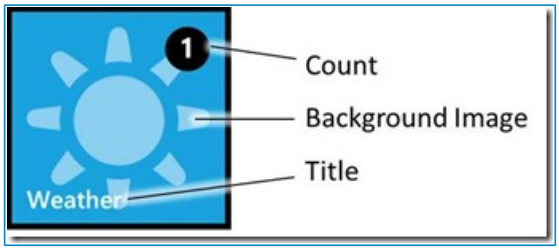
Push Notification的技术为什么越来越重要，其实这跟移动设备的特点紧密相关，移动设备网络的不稳定性，决定了以Socket为代表的强连接方式并不适用，所以大家更多选择HTTP协议作为主要的通讯方式。但是HTTP的特点是，设备找服务器很容易，通过URL就可以了，但服务器找设备就难上加难了，因为设备会随时切换移动网络，IP地址之类的经常性失效。当然设备端轮询的方式可以解决这个问题，但移动设备的电源、网络都是稀缺资源。所以，OS级别的Push Notification技术就变成了一种珍贵的战略资源，而且，在封闭式的操作系统中，只有OS厂商提供的Push Notification才能够获得最好的效率。

Push Notification简介

目前，Windows Phone支持三种Push Notification方式：Toast Notifications、Tile Notifications和Raw Notifications，我不想翻译成中文名字了，因为"吐司"之类的翻译无法帮助理解。

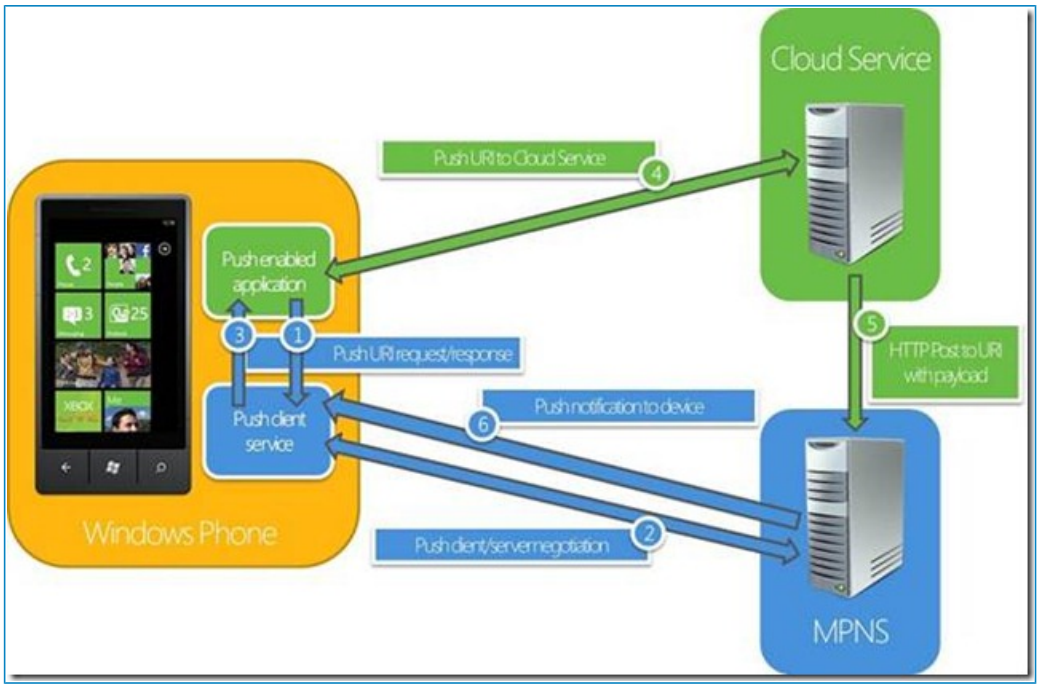
Toast Notifications，当我们的程序没有运行时，我们希望有一种形式可以通知用户，并且让用户调用对应的应用，就像收到SMS时，调用Messaging程序一样。运行效果如下图，当用户点击Toast时，可以调用对应的程序。

Tile Notifications，用于更新启动界面上的Tile，可以通过Tile Notifications设置Tile的背景图片、Count和Title等属性，各属性显示位置如下图所示。我们可以通过该技术来提示用户，我们的应用有新的事件发生，比如SNS上有多少新的回复等。



Raw Notifications，最简单，当你的应用程序运行时，可以接收Raw Notification信息，如果程序没有运行，则接受不到Raw Notification。

接下来，我们会以Toast Notifications为例详细介绍，Push Notification的原理和使用方法。



首先，还是引用这张广为流传的图示吧，确实是目前能够找到的最详细的示意图了。下面是步骤叙述：

1-3，应用通过调用HttpNotificationChannel接口，向Microsoft Push Notification Service (MPNS)请求一个URI，这个URI会在服务器端发送Notification消息时，作为验证发送目标的唯一标识；这样做的一个好处是，当同一个设备上有多个应用在监听Notification时，不会相互干扰。

4. 应用会将获得的URI传到自己的服务器上，这个URI会存储在自己的服务器上，用于发送Push Notification时调用。URI的传递和存储方式，由开发者自己决定，不过要保证传输过程的加密，以及存储时的安全性。接下来的实例里，传递和存储URI的方式就很有创意。

5-6. 当服务器端有信息要通知应用时，需要向MPNS发起一个Http请求，而MPNS将请求转发到相应的设备上，设备上的应用接到Push Notification的消息后，进行相应的处理。

其实过程挺简单，被MSDN文档说得神乎其神。接下来，我们挨个看一下各个类型的Notification。偷点懒，不自己写Sample Code了，请大家去下面的链接里下载。不过注意Sample Code的更新，最近有同学因为更新不及时，而影响了开发进度。

http://msdn.microsoft.com/en-us/library/ff431744(VS.92).aspx

初始化 Notifications

首先是Toast Notifications，这种Notification在实际应用中最为有效，不但可以提示用户有新的消息，而且还让用户选择是否打开当时的应用。不过比起iOS的Notifications来，Toast Notifications能够容纳的信息量实在有限。所以，开发者要适当"浓缩"需要推送的消息。

我们按照调用的顺序，依次来分析代码，首先来看客户端部分ToastNotificationClient的代码：

```
        /// Holds the push channel that is created or found.
        HttpNotificationChannel pushChannel;

        // The name of our push channel.
        string channelName = "ToastSampleChannel";

        InitializeComponent();

        // Try to find the push channel.
        pushChannel = HttpNotificationChannel.Find(channelName);

        // If the channel was not found, then create a new connection to the push service.
        if (pushChannel == null)
        {
            pushChannel = new HttpNotificationChannel(channelName);

            // Register for all the events before attempting to open the channel.
            pushChannel.ChannelUriUpdated += new EventHandler<NotificationChannelUriEventArgs>(PushChan
            pushChannel.ErrorOccurred += new EventHandler<NotificationChannelErrorEventArgs>(PushChanne

            // Register for this notification only if you need to receive the notifications while your
            pushChannel.ShellToastNotificationReceived += new EventHandler<NotificationEventArgs>(PushC

            pushChannel.Open();

            // Bind this new channel for toast events.
            pushChannel.BindToShellToast();

        }
        else
        {
            // The channel was already open, so just register for all the events.
            pushChannel.ChannelUriUpdated += new EventHandler<NotificationChannelUriEventArgs>(PushChan
            pushChannel.ErrorOccurred += new EventHandler<NotificationChannelErrorEventArgs>(PushChanne

            // Register for this notification only if you need to receive the notifications while your
            pushChannel.ShellToastNotificationReceived += new EventHandler<NotificationEventArgs>(PushC

            // Display the URI for testing purposes. Normally, the URI would be passed back to your web
            System.Diagnostics.Debug.WriteLine(pushChannel.ChannelUri.ToString());
            MessageBox.Show(String.Format("Channel Uri is {0}",
                pushChannel.ChannelUri.ToString()));

        }
```

代码比较多，我们逐步来看，首先要声明一个HttpNotificationChannel的变量，命名空间为Microsoft.Phone.Notification。接下来，声明该Notification Channel的名称，该名称是Channel的唯一标识名，不能与其他Channel重复。接下来，我们调用HttpNotificationChannel的静态方法Find来查找，当前的Channel是否存在，如果存在则返回HttpNotificationChannel对象实例，不存在则返回空。如果该程序从未在当前设备上运行过，则HttpNotificationChannel对象为空，只要运行过一次，则能够通过Find方法来获取对象实例。

先来说未创建Channel的情况，创建HttpNotificationChannel的对象，并将Channel名称作为参数传递进去。然后处理该对象的两个事件：ChannelUriUpdated和ErrorOccurred。当URI发生变化时，会触发ChannelUriUpdated事件，我们在创建Channel对象时，会触发该事件，另外一个ErrorOccurred事件，则用于错误处理。

当程序已经在运行时，默认是接收不到Toast Notification消息的，所以，我们还需要处理ShellToastNotificationReceived事件，当程序运行时，也可以接收到相应的消息。

然后调用HttpNotificationChannel的Open方法，打开该Channel，最后是BindToShellToast方法，通知Shell，该Notification的Channel已经被创建。

获取Channel后的代码与之大体类似，但我们可以通过HttpNotificationChannel的ChannelUri属性来获取Channel的URI，该URI值就是需要传递给我们自己服务器的唯一值，标记了该设备上的该Channel。

而创建Channel后，我们也可以获取URI的值，不过我们需要在ChannelUriUpdated事件处理函数里获取该URI值：

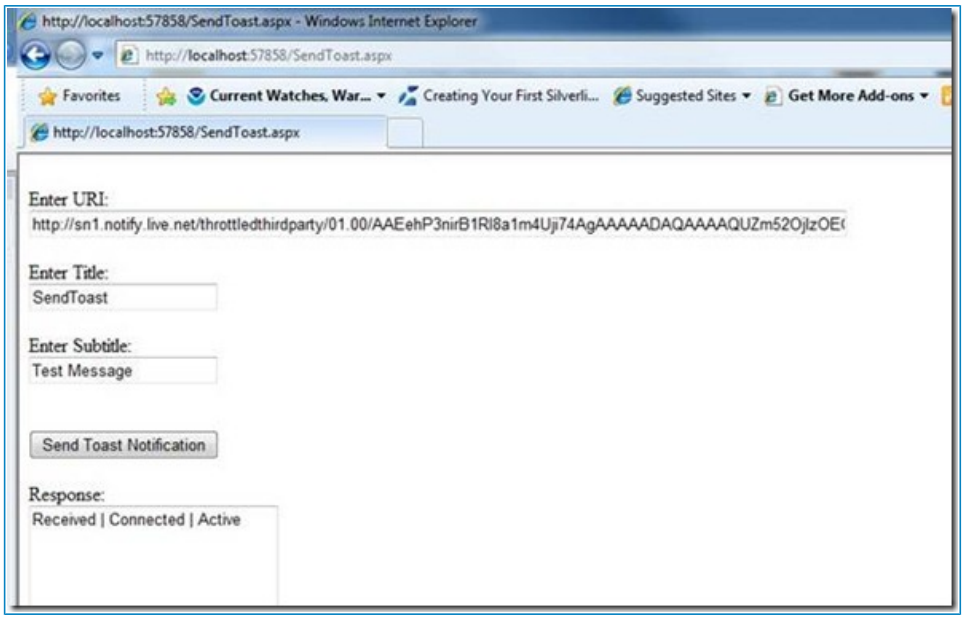
```
        void PushChannel_ChannelUriUpdated(object sender, NotificationChannelUriEventArgs e)
        {

            Dispatcher.BeginInvoke(() =>
            {
                // Display the new URI for testing purposes.   Normally, the URI would be passed back to yo
                System.Diagnostics.Debug.WriteLine(e.ChannelUri.ToString());
                MessageBox.Show(String.Format("Channel Uri is {0}",
                    e.ChannelUri.ToString()));

            });
        }
```

到此为止，我们已经完成了Notification Channel的创建，并且获得了唯一标识的URI。

触发Notification

接下来，我们来看Server的代码。我们自己的Server与Microsoft Push Notification Server（MPNS）之间是通过Http协议，使用HttpWebRequest来发送请求。理论上来说，我们也可以使用客户端程序向MPNS发送Notification请求。为什么要用ASP.NET来编写，最主要的原因还是在于客户端应用要将URI传递给Server端，显然基于ASP.NET的Web应用在这方面是最容易的。

不过我们的实例中，没有客户端将URI传递给服务器端的代码，而是采用了最简单的办法：由开发者直接将URI拷贝到ASP.NET的Web应用中。



接下来，我们来看Server端的代码：

```
        try
        {
            // Get the Uri that the Microsoft Push Notification Service returns to the Push Client when
            // Normally, a web service would listen for Uri's coming from the web client and maintain a
            // notifications out to.
            string subscriptionUri = TextBoxUri.Text.ToString();


            HttpWebRequest sendNotificationRequest = (HttpWebRequest)WebRequest.Create(subscriptionUri)

            // We will create a HTTPWebRequest that posts the toast notification to the Microsoft Push
            // HTTP POST is the only allowed method to send the notification.
            sendNotificationRequest.Method = "POST";

            // The optional custom header X-MessageID uniquely identifies a notification message.
            // If it is present, the // same value is returned in the notification response. It must be
            // sendNotificationRequest.Headers.Add("X-MessageID", "<UUID>");

            // Create the toast message.
            string toastMessage = "<?xml version=\"1.0\" encoding=\"utf-8\"?>" +
            "<wp:Notification xmlns:wp=\"WPNotification\">" +
               "<wp:Toast>" +
                    "<wp:Text1>" + TextBoxTitle.Text.ToString() + "</wp:Text1>" +
                    "<wp:Text2>" + TextBoxSubTitle.Text.ToString() + "</wp:Text2>" +
                    "<wp:Param>/Page2.xaml?NavigatedFrom=Toast Notification" + TextBoxSubTitle.Text.ToS
               "</wp:Toast> " +
            "</wp:Notification>";

            // Sets the notification payload to send.
            byte[] notificationMessage = Encoding.Default.GetBytes(toastMessage);

            // Sets the web request content length.
            sendNotificationRequest.ContentLength = notificationMessage.Length;
            sendNotificationRequest.ContentType = "text/xml";
            sendNotificationRequest.Headers.Add("X-WindowsPhone-Target", "toast");
            sendNotificationRequest.Headers.Add("X-NotificationClass", "2");


            using (Stream requestStream = sendNotificationRequest.GetRequestStream())
```

```
            using (Stream requestStream = sendNotificationRequest.GetRequestStream())
            {
                requestStream.Write(notificationMessage, 0, notificationMessage.Length);
            }

            // Send the notification and get the response.
            HttpWebResponse response = (HttpWebResponse)sendNotificationRequest.GetResponse();
            string notificationStatus = response.Headers["X-NotificationStatus"];
            string notificationChannelStatus = response.Headers["X-SubscriptionStatus"];
            string deviceConnectionStatus = response.Headers["X-DeviceConnectionStatus"];

            // Display the response from the Microsoft Push Notification Service.
            // Normally, error handling code would be here.  In the real world, because data connection
            // notifications may need to be throttled back if the device cannot be reached.
            TextBoxResponse.Text = notificationStatus + " | " + deviceConnectionStatus + " | " + notifi
        }
        catch (Exception ex)
        {
            TextBoxResponse.Text = "Exception caught sending update: " + ex.ToString();
        }
```

Server端程序的主体是调用HttpWebRequest对象，来实现HTTP POST的方法。发送的消息以XML形式呈现，在上面代码中表现为toastMessage字符串，wp:Toast指定Notification类型，wp:Text1指定标题，wp:Text2指定子标题，wp:Param指定传递的参数。

请大家注意wp:Param，我们可以认为该参数相当于调用了客户端应用中NavigationService的Navigate方法，必须确认客户端有指定的页面可以进行跳转。当用户点击Toast Notification的提示条时，客户端应用会调用wp:Param，从而跳转到指定界面上。

我们可以通过该参数来传递一些参数，比如标识本次Notification是由哪条消息引发的。不过该参数中不应该包括敏感信息、安全信息，因为无法保证这些信息的安全。最好的办法是，传递一个ID值，当客户端访问服务器时，由服务器端判断该ID是否有效。

接下来，我们还要为HTTP 头添加自定义字段，这部分参考Sample Code就可以了。

接收Notification

当Server端代码完成后，我们回到客户端，看一下当MPNS找到设备后，如何触发我们的客户端代码。首先是Page2.xaml的OnNavigatedTo方法：

```
    protected override void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)
    {
        base.OnNavigatedTo(e);

        //  If we navigated to this page
        // from the MainPage, the DefaultTitle parameter will be "FromMain".  If we navigated here
        // when the secondary Tile was tapped, the parameter will be "FromTile".
        textBlockFrom.Text = "Navigated here from " + this.NavigationContext.QueryString["NavigatedFrom
    }
```

通过NavigationContext.QueryString属性来获取自定义参数，试过传两个参数，后一个收到不到，目前成功的是传递一个参数的情况。

还有一种情况是，当客户端程序运行时，接收到Toast Notification时，该如何处理：

```
        void PushChannel_ShellToastNotificationReceived(object sender, NotificationEventArgs e)
        {
            StringBuilder message = new StringBuilder();
            string relativeUri = string.Empty;

            message.AppendFormat("Received Toast {0}:\n", DateTime.Now.ToShortTimeString());

            // Parse out the information that was part of the message.
            foreach (string key in e.Collection.Keys)
            {
                message.AppendFormat("{0}: {1}\n", key, e.Collection[key]);

                if (string.Compare(
                    key,
                    "wp:Param",
                    System.Globalization.CultureInfo.InvariantCulture,
                    System.Globalization.CompareOptions.IgnoreCase) == 0)
                {
                    relativeUri = e.Collection[key];
                }
            }

            // Display a dialog of all the fields in the toast.
            Dispatcher.BeginInvoke(() => MessageBox.Show(message.ToString()));

        }
```

该方法是HttpNotificationChannel的ShellToastNotificationReceived事件处理函数，具体内容不详细解释了。

Tile Notifications

Tile Notification是另一种Notification，主要用于在Windows Phone的首页上显示Smart Tile。很多内置的WP应用可以显示Tile上的自定义显示，比如Messaging程序里可以显示当前未读的短信有多少。Tile Notification可以帮助我们实现类似的功能，通常的一个用法是，首先发送一条Toast Notification，如果用户不理睬，可以发送一条Tile Notification，在Smart Tile上显示提示，让用户在方便时，再处理该条请求。

Tile Notification的实现代码与Toast Notification类似，所以，我们不在这里列出全部源代码了，只是列出与上面不同的部分代码。

首先来看Server部分的代码：

```
            // Create the tile message.
            string tileMessage = "<?xml version=\"1.0\" encoding=\"utf-8\"?>" +
            "<wp:Notification xmlns:wp=\"WPNotification\">" +
                "<wp:Tile>" +
                  "<wp:BackgroundImage>" + TextBoxBackgroundImage.Text + "</wp:BackgroundImage>" +
                  "<wp:Count>" + TextBoxCount.Text + "</wp:Count>" +
                  "<wp:Title>" + TextBoxTitle.Text + "</wp:Title>" +
                  "<wp:BackBackgroundImage>" + TextBoxBackBackgroundImage.Text + "</wp:BackBackgroundIm
                  "<wp:BackTitle>" + TextBoxBackTitle.Text + "</wp:BackTitle>" +
                  "<wp:BackContent>" + TextBoxBackContent.Text + "</wp:BackContent>" +
               "</wp:Tile> " +
            "</wp:Notification>";

            // Sets the notification payload to send.
            byte[] notificationMessage = Encoding.Default.GetBytes(tileMessage);

            // Sets the web request content length.
            sendNotificationRequest.ContentLength = notificationMessage.Length;
            sendNotificationRequest.ContentType = "text/xml";
            sendNotificationRequest.Headers.Add("X-WindowsPhone-Target", "token");
            sendNotificationRequest.Headers.Add("X-NotificationClass", "1");
```

TileMessage与ToastMessage大体类似，标签变为wp:Tile，而其中包括的数据分为两组：BackgroundImage、Title和Content，显示位置可以参考文章前面的示意图。BackgroundImage可以是本地的资源，也可以是远程的图片链接。

接下来是客户端的部分代码：

```
            pushChannel = new HttpNotificationChannel(channelName);

            // Register for all the events before attempting to open the channel.
            pushChannel.ChannelUriUpdated += new EventHandler<NotificationChannelUriEventArgs>(PushChan
            pushChannel.ErrorOccurred += new EventHandler<NotificationChannelErrorEventArgs>(PushChanne

            pushChannel.Open();

            // Bind this new channel for Tile events.
            pushChannel.BindToShellTile();
```

Tile Notification无需添加任何额外的代码，在调用HttpNotificationChannel的Open方法后，还需要调用BindToShellTile的方法，通知Shell绑定该Tile Notification。

Raw Notifications

Raw Notification的用法最简单，在程序运行时，收到Raw Notification时，更新应用程序界面上的某些元素。

首先，我们来看Server端的代码：

```
            // Create the raw message.
            string rawMessage = "<?xml version=\"1.0\" encoding=\"utf-8\"?>" +
            "<root>" +
                "<Value1>" + TextBoxValue1.Text.ToString() + "<Value1>" +
                "<Value2>" + TextBoxValue2.Text.ToString() + "<Value2>" +
            "</root>";

            // Sets the notification payload to send.
            byte[] notificationMessage = Encoding.Default.GetBytes(rawMessage);

            // Sets the web request content length.
            sendNotificationRequest.ContentLength = notificationMessage.Length;
            sendNotificationRequest.ContentType = "text/xml";
            sendNotificationRequest.Headers.Add("X-NotificationClass", "3");
```

RawMessage非常简单，只需要指定Value1, Value2就可以了，这些值是自定义的，然后指定HTTP头的X-NotificationgClass的值为3，代表Raw Notification。

接下来是客户端代码的片段：

```
pushChannel = new HttpNotificationChannel(channelName);
pushChannel.HttpNotificationReceived += new EventHandler<HttpNotificationEventArgs>(PushChannel_HttpNotific
pushChannel.Open();


        void PushChannel_HttpNotificationReceived(object sender, HttpNotificationEventArgs e)
        {
            string message;

            using (System.IO.StreamReader reader = new System.IO.StreamReader(e.Notification.Body))
            {
                message = reader.ReadToEnd();
            }


            Dispatcher.BeginInvoke(() =>
                MessageBox.Show(String.Format("Received Notification {0}:\n{1}",
                    DateTime.Now.ToShortTimeString(), message))
                    );
        }
```

HttpNotificationChannel创建完成后，只需要调用Open方法即可，不需要绑定到Shell上。另外，还需要处理HttpNotificationChannel的HttpNotificationReceived事件处理函数，具体的代码不解释了。

写在最后

好了，到这里，我们正式将Windows Phone 7.1中的Push Notification介绍完了。为了增加用户黏性，Push Notification可能是一种非常有效的方式，但是请开发者谨慎使用，如果太多的垃圾信息影响用户体验，那是得不偿失的。另外，我们还需要为Push Notification搭建一个Web服务器，这对于普通开发人员来说，也是一个很难接受的成本。

OpenXLive杯Windows Phone游戏开发大赛

OpenXLive杯Windows Phone游戏开发大赛，是由OpenXLive联合国内知名的开发者社区：DevDiv、智机网、WPMind、Silverlight银光中国和XNA游戏世界，一起举办的针对Windows Phone游戏开发的比赛。

http://www.openxlive.net/posts/news/40