Windows 8 开发 31 日

第 12 日

后台任务

译者：BeyondVincent(破船)

时间：2013.4.25

版本： 2.0

关于破船

程序猿砌墙于云南昆明!

长期扎根移动软件开发!

爱跑步爱打篮球爱运动!

命中无大富大贵之面相!

愿健康与平淡相随一生!

你可以发邮件与破船取得联系: BeyondVincent@gmail.com

还可以关注破船的微博: 腾讯微博和新浪微博。

这里是破船的个人博客，欢迎光临：破船之家

关于 Windows 8 开发 31 日翻译

Windows 8 开发 31 日是由 Jeff Blankenburg 和 Clark Sell 原创的。

官方站点：http://31daysofwindows8.com/

涉及到两个版本：

XAML/C#（由 Jeff Blankenburg 撰写）

HTML5/JS （由 Clark Sell 撰写）

其中涉及到的资源和相关代码请到这里下载：

https://github.com/csell5/31DaysOfWindows8

在这里，由于破船对 HTML5/JS 不熟悉，所以只翻译 XAML/C#相关主题。

建议大家前往看原创内容，如果看不明白，再来这里看我翻译的相关内容。

如果翻译不正确的地方，可以通过上面的联系方式告诉破船。

破船祝你阅读愉快！

目录

第 12 日后台任务

1.0. 介绍

今天我们来讨论一下后台任务。简单的来说，后台任务是：

后台任务是当程序没有运行时所运行的代码。

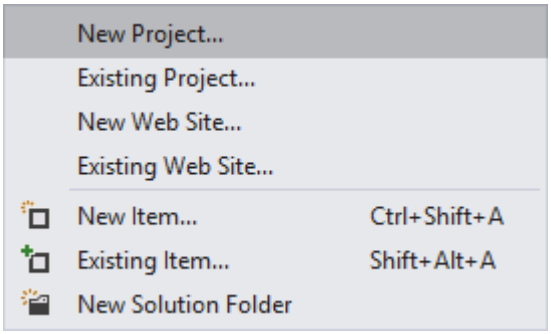
想想下面的场景：

- 当用户运行别的程序是，持续播放音乐
- 更新用户动态磁贴以反映新数据
- 弹出一个 Toast 通知，告诉用户发生了一些重要的事情
- 当设备锁定的时候，更新用户的锁屏

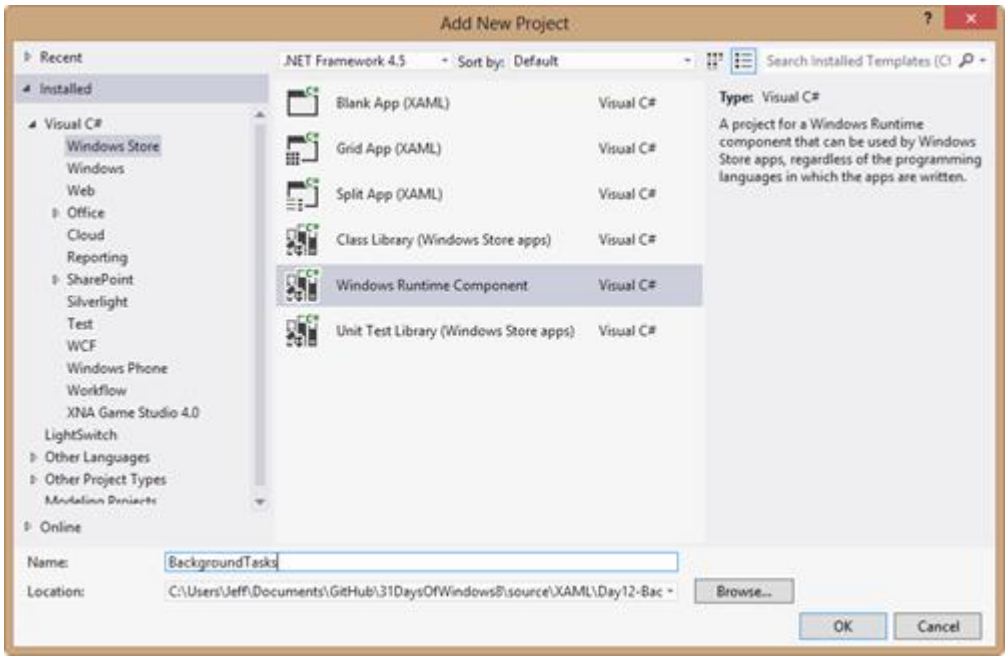
在系统里面创建和注册一个后台任务有两个步骤。首先，必须在 package.appxmanifest 文件中注册一个后台任务。注册之后，还需要在程序代码中向系统注册任务的触发事件，然后当后台任务结束时，还有额外的事件来做相关的管理。本文会涉及到上面提到的这些内容。

1.1. 创建一个后台任务类

在这里，除了创建一个普通的 Blank App 外，我还创建另外一个工程，用来编写我的后台任务。当然，这不是必须的，不过这样做有利于代码功能的分离：



在创建一个新的工程时，选择“Windows Runtime Component”作为工程类型。



默认情况，会给定一个名为 Class1.cs 的类。你可以修改这个名字，删除它并

添加一个新的类到工程中。在这里，我创建一个类文件，命名为 TileUpdater.cs。

现在，后台任务，必须实现一个特殊的接口：IBackgroundTask。该接口中只有一个方法需要直线：Eun()，当我们的程序启动了后台任务时，这个方法会被调用。下面是我用 C#实现的接口：

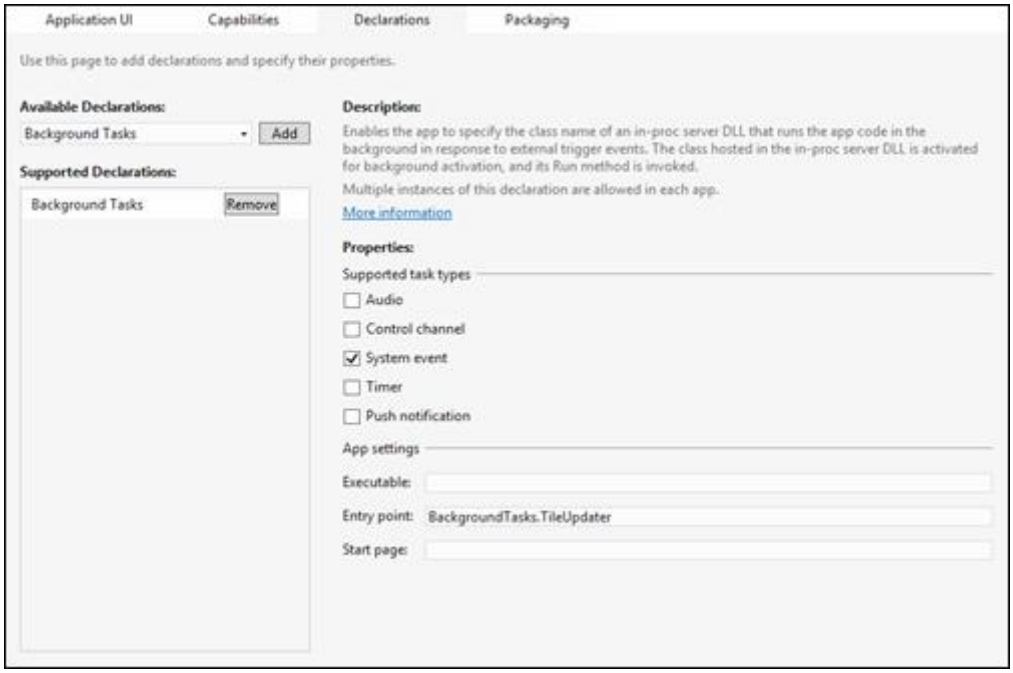
```
using System;
usingSystem.Collections.Generic;
usingSystem.Linq;
usingSystem.Text;
usingSystem.Threading.Tasks;
usingWindows.ApplicationModel.Background;
usingWindows.Data.Xml.Dom;

namespaceBackgroundTasks
{
publicsealedclassTileUpdater : IBackgroundTask
    {
publicvoid Run(IBackgroundTaskInstancetaskInstance)
        {
// we will add code here later.
        }
    }
}
```

我在代码中添加了一个注释，对于现在，它是一个好的开始。

1.2. 设置 Packge.Appxmanifest 文件

下一步就是声明在程序中要后台任务。声明是在 package.appxmanifest 文件中进行，在 Declaration 选项卡中，如下图所示进行设置（点击看大图）：



如上所示，我添加了一个 Background Task 声明（每一个后台任务都需要工程中添加一个这样的声明），并将 Entry point 设置为 BackgroundTasks.TileUpdater，这是我新建的一个文件。下一步，需要在实际的程序中注册这个 task，那样 Windows 8 才知道在适当的时机运行 task。

1.3. 注册后台任务

下面我们需要在 MainPage.xaml.cs 文件中做几步操作，首先是检查后台任务是否已经在系统里面注册了。一般我是在页面加载的时候检查的，也就是 OnNavigatedTo 方法中。我创建了一个新的 CheckTaskRegistration()方法来检查任务的注册状态，并设置一个页面内的全局 Boolean 值，你也可以在 App.xaml.cs 文件中进行检查，不过在这里为了简单，我在 MainPage.caml.cs 中进行检查。

```
boolisTaskRegistered = false;
```

```
protectedoverridevoidOnNavigatedTo(NavigationEventArgs e)
{
CheckTaskRegistration();
}
```

```
privatevoidCheckTaskRegistration()
{
foreach (var task inBackgroundTaskRegistration.AllTasks)
        {
if (task.Value.Name == "TileUpdater")
            {
isTaskRegistered = true;
break;
            }
        }

if (isTaskRegistered)
        {
RegisterButton.Content = "Unregister Background Task";
        }
elseif (!isTaskRegistered)
        {
RegisterButton.Content = "Register Background Task";
        }
}
```

正如你说看到的，我需要对 BackgroundTaskRegistration.AllTasks 进行循环处理，因为在程序中，有可能有多个后台任务在运行着，在这里，我只对 TileUpdater 感兴趣。在后面的代码，你可以看到，如果当前没有注册任务，我会调用 RegisterBackgroundTask()方法注册一下。

至于后台任务什么时候注册,这取决你，在我的示例程序中，我希望后台任务一直都是注册的，如果由于某些原因取消注册了，我希望再次添加注册。下面是我的 RegisterBackgroundTask()方法。

```
privatevoidRegisterBackgroundTask(string name, stringentrypoint)
```

```
{
BackgroundTaskBuilderbtb = newBackgroundTaskBuilder();
btb.Name = name;
btb.TaskEntryPoint = entrypoint;
btb.SetTrigger(newSystemTrigger(SystemTriggerType.InternetAvailable, false));

BackgroundTaskRegistration task = btb.Register();

task.Progress += task_Progress;
task.Completed += task_Completed;
}
```

可以看到，上面的方法中我传递了 name 和 entrypoint 值进去，当然，你也可以在这里进行硬编码，这取决于你的具体情况。因为这个程序可能有多个后台任务，所以我我将注册方法抽取出来，通过传递相关参数进行注册。

在我的注册方法中，我创建了一个 BackgroundTaskBuilder 对象，并设置它的 name 和 entry point，然后用一个 SystemTrigger 来触发后台任务的运行。在这里我选择了容易测试的 InternetAvailable 来触发。为了测试这个触发，我会首先机器的切断网络访问。

如果你还没有取消注册你的后台任务，那么最好写一下吧，很简单的。下面是示例中的方法，UnregisterBackgroundTask()方法可以移除后台任务。

```
privatevoidUnregisterBackgroundTask(string name)
{
foreach (var task inBackgroundTaskRegistration.AllTasks)
    {
if (task.Value.Name == name)
        {
task.Value.Unregister(true);
        }
    }
}
```

在示例中，我循环枚举已经注册的后台任务，如果发现与我想要找的名称一

样的任务，我将调用 Unregister()方法，并传入 true 方法。

Ok，现在已经完成后台任务的注册和取消注册。下面我们来给后台任务写一些实际的代码，然后再测试一下吧。

1.4. 让后台任务做一些事

之前，我在我新创建的 Windows Runtime Component 工程中添加了一个名为 TileUpdater.cs 的文件，在该文件中，必须要实现 IBackgroundTask 接口的 Run()方法。下面是我在 Run()方法中添加的更新动态磁贴的代码。

```
using System;
usingSystem.Collections.Generic;
usingSystem.Linq;
usingSystem.Text;
usingSystem.Threading.Tasks;
usingWindows.ApplicationModel.Background;
usingWindows.Data.Xml.Dom;
usingWindows.UI.Notifications;

namespaceBackgroundTasks
{
publicsealedclassTileUpdater : IBackgroundTask
    {
publicvoid Run(IBackgroundTaskInstancetaskInstance)
        {
XmlDocumenttileData = TileUpdateManager.GetTemplateContent(TileTemplateType.TileSquareText04);
XmlNodeListtextData = tileData.GetElementsByTagName("text");
textData[0].InnerText = "Background updates are absolutely amazing. #31daysofwin8";
TileNotification notification = newTileNotification(tileData);
notification.ExpirationTime = DateTimeOffset.UtcNow.AddSeconds(30);
TileUpdateManager.CreateTileUpdaterForApplication().Update(notification);
        }
    }
}
```

很明显，我们也可以写类似的后台任务来实现发送 Toast 通知，更新锁屏信息，或者其它可以调用的 web services。

1.5. 总结

关于后台任务，这里有一个事情需要注意：后台任务的调试要比在工程中的代码难。实际上，微软已经写了一篇关于如何调试后台任务的文章。强烈建议阅读一下。微软同样写了快速开始的教程：Create and Register a Background Task。

点击下面的图标，可以下载本文的示例代码：

明天，我将开始介绍新的内容：关于导航（Navigation）。具体涉及到在 XAML 程序中，如何在两个 page 之间进行导航，以及在导航的时候如何传递参数和数据。到时候见！

感谢你的阅读！

如果对这篇文章有什么想法想法，可以与破船联系，破船的联系方式在文章开头。

破船