Windows 8 开发 31 日

第 29 日

应用程序的生命周期

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。

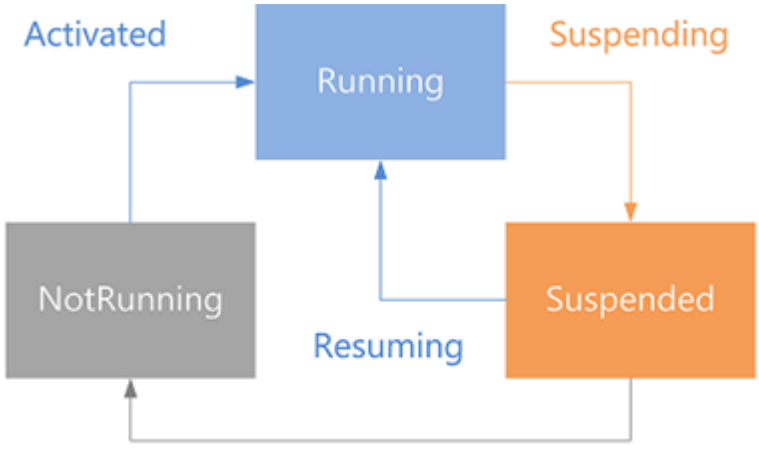
破船祝你阅读愉快！

目录

第 29 日应用程序的生命周期

1.0. 介绍

今天我将介绍应用程序的生命周期。针对这个主题我希望有一个很酷的词语，但是最终我们将学习在用户设备中程序里面不同状态的管理。下面是一个演示图（图片来自 MSDN）。



本部分内容针对 Windows 8 应用开发者来说是非常重要的，因为我们希望在程序中能够给用户提供一个无缝的用户体验。当用户按下设备上的 Windows 键时，会离开你的程序，此时你需要确保用一些方法将程序中的某些数据保存起来，这样当再次回到你的程序时，这些数据还能够还原出来。

1.1. 我认为 windows 会自动处理程序的生命周期

是和不是。默认情况下，如果你运行一个 Windows 应用，然后离开这个程序，之后再次回到程序，Windows 会准确的做出相应的处理。不过，如果离开了你的程序，然后再打开 6-7 个别的程序，操作系统会把你的程序进入休眠，甚至是终止程序。很好的是，在系统中，有一个很棒的机制，我们可以利用这个机制来做一些管理。

1.2. 识别程序进入休眠（suspended）状态

根据微软的说法“当用户离开程序时，或者 Windows 系统进入低电状态，程序可以被休眠。觉用户离开程序时，大多数的程序都会停止运行。”下面有一些事情你需要知道——在休眠状态中的程序如何会被终止：

你的程序“可能”会被休眠。在本文我写的这个程序，我从来没有见过它会进入休眠状态。

- 通常，当系统耗尽资源时，你的程序会进入休眠状态。
- 当你离开一个程序，进入另外一个程序时，在离开的那个程序进入休眠之前，Windows 会等待几秒钟，以防你立马就返回。
- 在程序进入休眠之前，如果你需要执行任意的代码（下面我会介绍），那么在程序没有反应和被终止之前，你有 5 秒的时间来执行代码。切记这点…5 秒钟 web service 的调用可能不会有任何结果。

如果你跟我一样，你可能也希望来写一些代码测试一下这种情况是如何发生

的。很好的是，这里有一些工具可以使用。

首先，我们来写一些代码。在每个画面中，用户可能会提供一些内容（不管是什么类型的数据），我们可以添加一个 event handler，这样当程序被休眠时，这个 handler 会被调用。在下面的代码中，我创建了一个 event handler，当被调用的时候，我将两个值保存到本地存储中。

```
using System;
using Windows.Storage;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Navigation;

namespace Day29_ApplicationLifecycle
{
    public sealed partial class MainPage : Page
    {
        public MainPage()
        {
            this.InitializeComponent();
        }

        ApplicationDataContainer settings;

        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
            Application.Current.Suspending += Current_Suspending;
            settings = ApplicationData.Current.LocalSettings;
        }

        void Current_Suspending(object sender, Windows.ApplicationModel.SuspendingEventArgs e)
        {
            settings.Values["suspendedDateTime"] = DateTime.Now.ToString();
            settings.Values["customTextValue"] = CustomText.Text;
        }

        protected override void OnNavigatingFrom(NavigatingCancelEventArgs e)
        {
            Application.Current.Suspending += Current_Suspending;
        }
    }
}
```

如上代码，我创建的 event handler 跟别的也一样，当 OnNavigatingFrom event 被调用的时候，我同样将这个 handler 移除掉。第 8 日中我介绍如了数据的保存，因此，在这里我不会对数据保存进行介绍。在上面的代码中，我把程序休眠的日期保存起来，另外还保存了用户在 MainPage.xaml 页面中 TextBox 输入的任意文本内容。为了验证保存的这些信息，下面我来添加一个恢复（Resuming）事件。

1.3. 识别程序程序进入恢复（resuming）状态

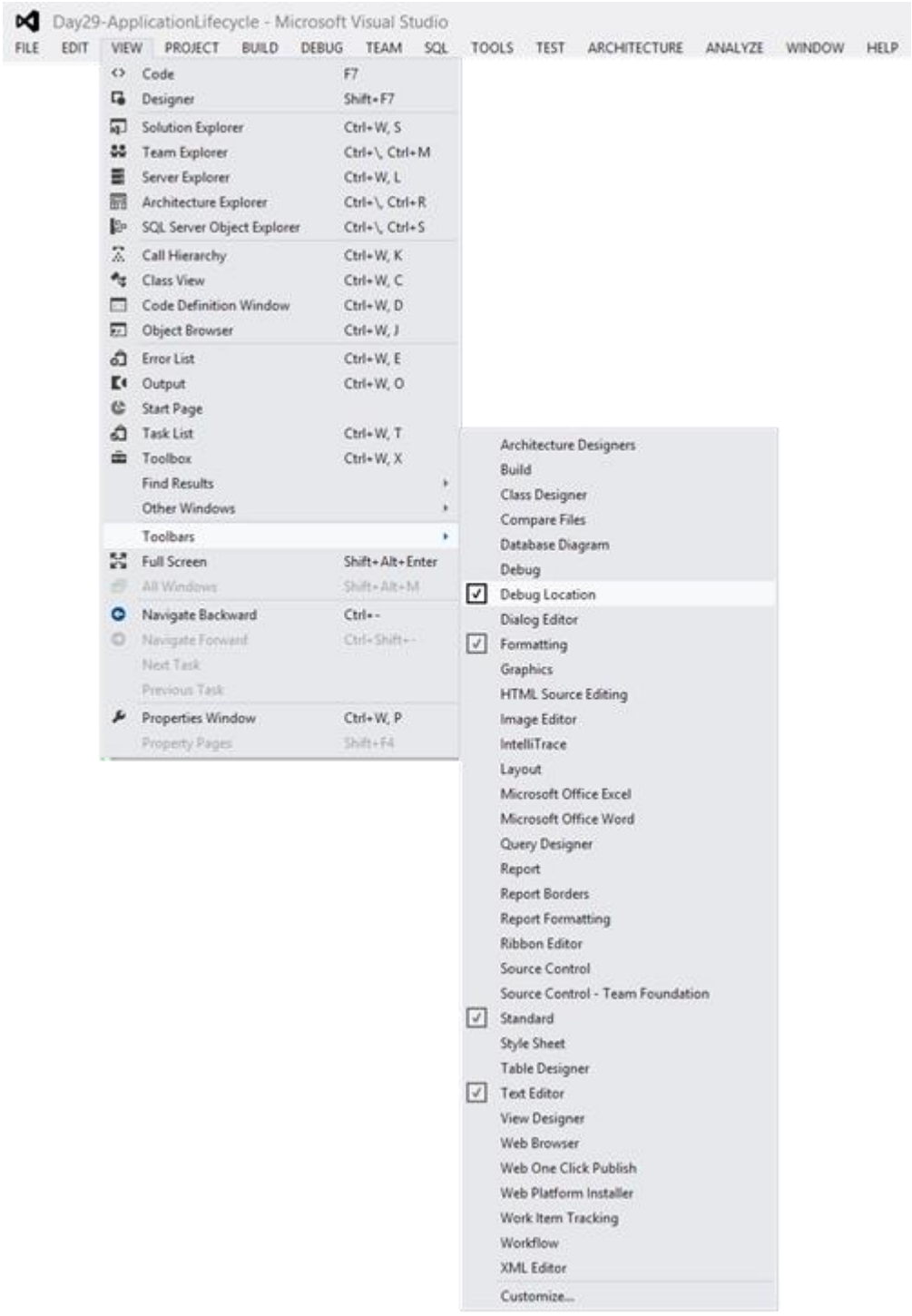
在这里，我将添加一个恢复（Resuming）事件，操作跟休眠事件类似。方法被调用的时候，我将把日期和文本内容显示到屏幕中，这样，用户就可以继续利用之前输入的数据进行相关操作。

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using Windows.Foundation;
using Windows.Foundation.Collections;
using Windows.Storage;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Controls.Primitives;
using Windows.UI.Xaml.Data;
using Windows.UI.Xaml.Input;
using Windows.UI.Xaml.Media;
using Windows.UI.Xaml.Navigation;

namespace Day29_ApplicationLifecycle
{
    public sealed partial class MainPage : Page
    {
        public MainPage()
        {
            this.InitializeComponent();
        }

        ApplicationDataContainer settings;

        protected override void OnNavigatedTo(NavigationEventArgs e)
```

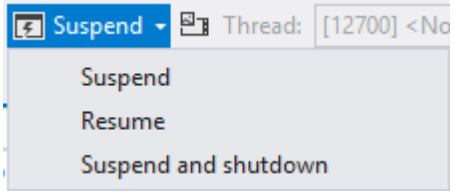
```
        {
            Application.Current.Suspending += Current_Suspending;
            Application.Current.Resuming += Current_Resuming;
            settings = ApplicationData.Current.LocalSettings;
        }

        void Current_Suspending(object sender, Windows.ApplicationModel.SuspendingEventArgs e)
        {
            settings.Values["suspendedDateTime"] = DateTime.Now.ToString();
            settings.Values["customTextValue"] = CustomText.Text;
        }

        void Current_Resuming(object sender, object e)
        {
            Message.Text = "Resumed.    Was suspended at\n\n" + settings.Values["suspendedDateTime"];
            CustomText.Text = settings.Values["customTextValue"].ToString();
        }

        protected override void OnNavigatingFrom(NavigatingCancelEventArgs e)
        {
            Application.Current.Resuming -= Current_Resuming;
            Application.Current.Suspending += Current_Suspending;
        }

    }
}
```

1.4. 在实际开发中如何测试这些事件呢？

由于要让这些状态（休眠和恢复）自动发生是很难的，不过在 Visual Studio 中有相应的工具可以让我们在程序中模拟这些状态。首先，确保打开了 Visual Studio 中的 **Debug Location** 工具栏。



当添加了这个工具栏后，就可以很方便的选择这些状态了：



- 休眠（**Suspend**）会立即将程序设置为休眠状态。在程序停止运行之前，还会触发你的休眠 event handler。
- 恢复（**Resume**）会将程序从休眠状态恢复出来。如果程序没有休眠，则不会发生任何事情。
- 休眠并关闭（**Suspend and shutdown**）会模拟 Windows 终止你的程序。首先会发生休眠状态，接着是完整的关闭你的程序。

简短的来说，上面介绍的是基于页与页之间的数据管理。如果你想要在程序级别上做一些事情，那么你可以看看 App.xaml.cs 文件。

1.5. 使用 App.xaml.cs 来管理生命周期

在 App.xaml.cs 文件中，这里已经有两个方法了，在本系列文章的之前，你已经使用过它们了：

OnLaunched：当程序正常启动时，会调用这个方法。当程序被恢复时，这个方法不会被调用，而是直接恢复到用户之前浏览的页面。

OnSuspending：在页面 event 调用之后，会调用这个方法，当然，程序被休眠的时候，也会调用这个方法。程序级别的数据保存就应当在这个方法中进行。

虽然这两个方法提供了一个全局保存数据的功能，不过大多数状态的保存还

是需要依赖于页面级别的事件。

1.6. 总结

简单来说，如果我们想要为提供无缝的用户体验，那么请记住本文的内容吧。但也不要过渡使用本文介绍的内容。为用户保存尚未保存的数据是很好的，但是不要把所有的内容都在这里保存。

在游戏中，如果用户离开了，而用户还需要回到想要的级别上，那么久将这个级别保存起来。你不需要记住用户已经跳过某个方块，以加强给好的用户体验，只需要记住级别就够了。

点击下面的图片，可以下载本文相关示例代码：

明天，我将介绍 Windows Store，并在明天的文中，介绍让程序能够在 Windows Store 中销售所需要的所有信息。到时候见吧 ！

感谢你的阅读！

如果对这篇文章有什么想法，可以与破船联系，破船的联系方式在文章开头。

破船