Windows 8 开发 31 日

第 08 日

本地和漫游数据

译者：BeyondVincent(破船)

时间：2013.4.23

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 08 日本地和漫游数据

1.0. 介绍

在本系列的好几篇文章,我都提到了存储数据非常重要，并且实现起来非常简单，包括本地设备存储，以及在不同设备之间漫游。

在使用漫游 VS.本地存储时，微软给我们提供了具体的指导，下面我对这个指导进行了一个总结。当然，如果你打破了这些规则，没有遵守这些指导，也不会被拒绝，不过在数据传输的大小和速度会用限制，超过这些限制的话，将会阻止你的程序获取实际的漫游数据。

DO

首选项设置和定制的数据可以使用漫游。用户可能希望它们选择的设置可以在每台机器上都相同。包括基本的设置 ,比如颜色、喜好、是否将数据发布到 Twitter 等。

漫游用户的某项工作。比如运行用户可以接着写没有完成的 email 或者 blog。

DON'T

仅仅与本地相关的信息则不要使用漫游。包括文件路径和其它一些至于本地

机器相关的数据信息。

不要漫游大型数据。最好只漫游首选项和小量数据文件。可以在代码中限制漫游数据的尺寸。

不要漫游及时同步或者快速变化的数据。Windows 控制着你的数据多久被漫游一次，所以别要考虑需要及时同步的数据。如果你需要及时同步的话建议创建自己的一个 web service。同样，也不要频繁的更新漫游数据。例如，你不需要总是一直更新用户当前的位置信息，而应该每隔几分钟或者更多。

最后就是请记住 漫游与设备之间的数据是通过用户的 Microsoft 账号管理的。如果用户用相同的账号登录了两个设备，并且在两个设备上安装了相同的程序，则会开始漫游，直到没有需要漫游的数据。

现在我有点担心你从来不是是用本地和漫游数据，下面我就来介绍一下是如何使用的。这里有两种类型的数据能够被存储，我在这里将每种数据的本地存储和漫游。首先从设置信息开始，然后是文件。

1.1. 设置信息的本地存储和漫游

在 Windows 8（或 Windows Phone）开发中，当你听到“设置”这个词时，应该立马就想到“小而简单的数据”。我们讨论的就是关于 name/value 的存储。

用户首选项就是一个很好的例子，在你的游戏中，存储用户的名字（字符串），可能用户关闭通知（boolean 值）。设置也是也是很容易存储的数据，在我写的程序

中,存储了设置信息值。因为这些值是看不到的，所以我在示例中将许多方法调用连接在一起:创建、读取和删除设置值，包括本地存储和漫游存储。你会发现，实际上使用非常简单。

```
ApplicationDataContainer settingsLocal;
ApplicationDataContainer settingsRoaming;
string currentBook;
int currentPage;
```

```
public MainPage()
{
this.InitializeComponent();

        settingsLocal = ApplicationData.Current.LocalSettings;
        settingsRoaming = ApplicationData.Current.RoamingSettings;

        AddSettings();
}
```

```
privatevoid AddSettings()
{
//There is no reason to set the same data to both local and roaming.
//This is here merely for illustration of HOW to do it.
//You should make the choice as to whether your data should be roamed.

        settingsLocal.Values["currentBook"] = "Hitchhiker's Guide To The Galaxy";
        settingsLocal.Values["currentPage"] = 42;

        settingsRoaming.Values["currentBook"] = "Hitchhiker's Guide To The Galaxy";
        settingsRoaming.Values["currentPage"] = 42;

        ReadSettings();
}
```

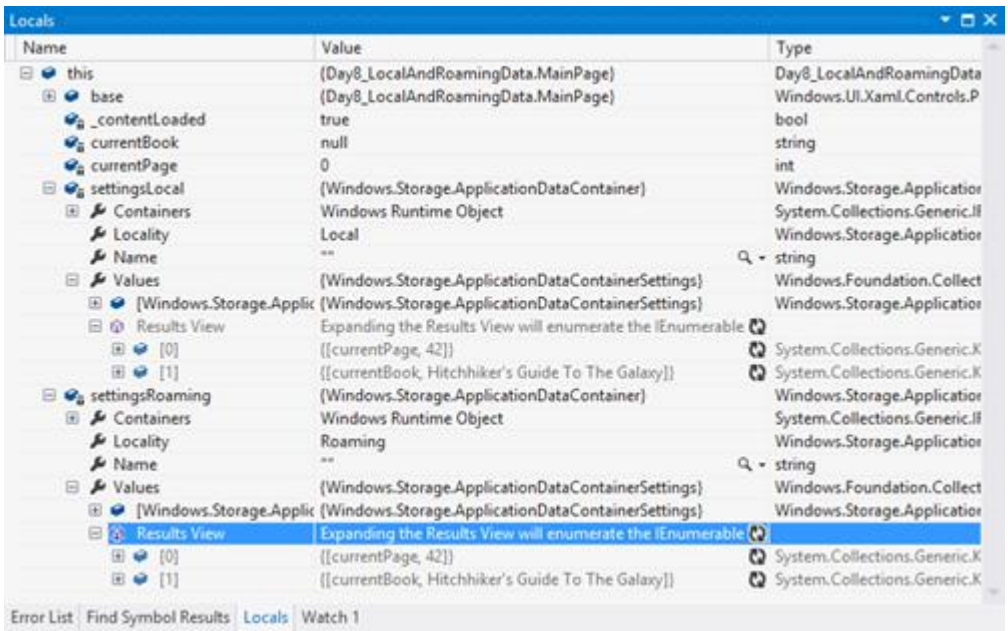
```
privatevoid ReadSettings()
{
//If you want typed data when you read it out of settings,
//you're going to need to know what it is, and cast it.

        currentBook = (string)settingsLocal.Values["currentBook"];
        currentPage = (int)settingsRoaming.Values["currentPage"];

        DeleteSettings();
```

```
}
```

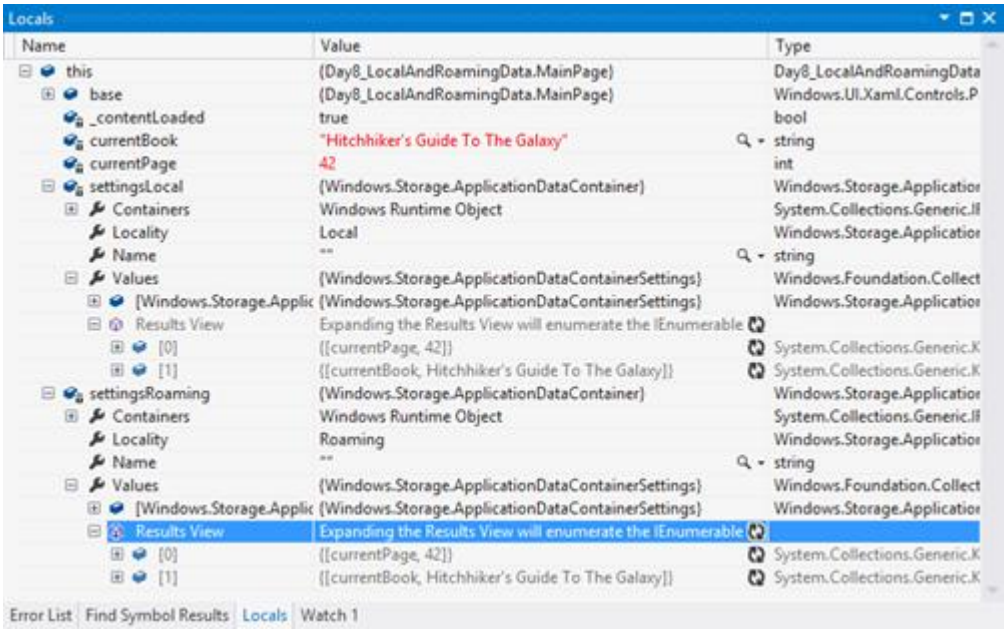
```
privatevoid DeleteSettings()
{
        settingsLocal.Values.Remove("currentBook");
        settingsLocal.Values.Remove("currentPage");

        settingsRoaming.Values.Remove("currentBook");
        settingsRoaming.Values.Remove("currentPage");

        AddCategoryAndNewSettingValue();
}
```

在工程中，你可以在每个函数的开始处设置断点，然后在 Locals 选项卡中使用 Visual Studio 检查设置值。

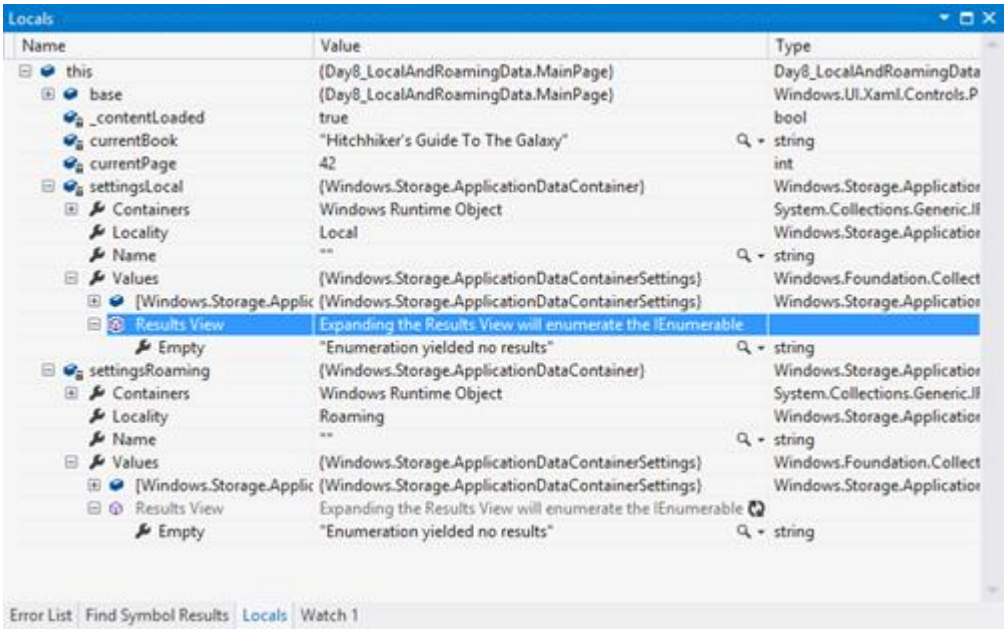
下面是 AddSettings()方法执行后的结果（单击看大图）：



当从从设置中读取数据后，存储在里面的内容如下：



最后，DeleteSettings()方法执行完毕时：



当你在构建程序时,需要记住的一个重点是所有这些数据,包括设置和文件,都以沙箱的形式存储。也就是说，当你卸载程序时，所有这些数据都将被删除。

另外在宝春这些 name/value 时，可能你想将它们分类。同样，我们可以在设置中创建分类——可以很简单的在设置中添加和移除分组。

同样本地和漫游都支持分组。下面，我创建了一个分类，并在该分类下添加一个值。

```
privatevoid AddCategoryAndNewSettingValue()
{
    settingsLocal.CreateContainer("mediaSettings", ApplicationDataCreateDisposition.Always);
    settingsLocal.Containers["mediaSettings"].Values["Volume"] = 11;
}
```

现在，我们已经深入了解了设置的存储,下面我们该学习文件了。

1.2. 文件的本地存储和漫游

关于文件的操作步骤除了读写而外，与设置类似。下面我将演示该示例。我创建了一组与上一节中设置相关类似的方法：AddFile()，ReadFile()和 DeleteFile()。下面来看看具体代码：

```
StorageFolder folderLocal;
StorageFolder folderRoaming;
string fileName = "tacotext.txt";
string fileContents = "taco";
```

```
public MainPage()
{
this.InitializeComponent();

    folderLocal = ApplicationData.Current.LocalFolder;
    folderRoaming = ApplicationData.Current.RoamingFolder;

    AddFile();
}
```

```
privateasyncvoid AddFile()
{
StorageFile fileLocal = await folderLocal.CreateFileAsync(fileName, CreationCollisionOption.ReplaceExisting);
awaitFileIO.WriteTextAsync(fileLocal, fileContents + "taco");

      ReadFile();
}
```

```
privateasyncvoid ReadFile()
{
StorageFile fileLocal = await folderLocal.GetFileAsync(fileName);
string textLocal = awaitFileIO.ReadTextAsync(fileLocal);

      fileContents = textLocal;

      DeleteFile();
}
```

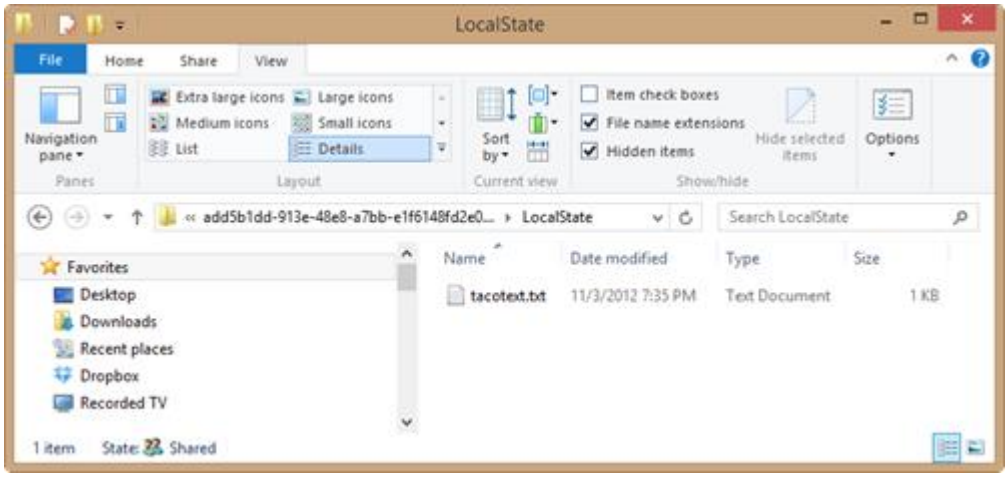
```
privateasyncvoid DeleteFile()
{
StorageFile fileLocal = await folderLocal.GetFileAsync(fileName);
      fileLocal.DeleteAsync();
}
```

如上代码，主要的不同就是 async/await 相关的读写的方法。我们没必要指定文件夹的位置，也没必要定义文件夹的结构。

另外,你可以查看文件存储的位置。每个程序将它的文件存储在设备中，如果你设置断点 ,则可以看到文件所在设备中的位置。例如 ,下面是我创建的 tacotext.txt 文件存储的 Path 属性：

当你创建好文件之后，就可以通过文件浏览器定位到该路径，甚至还可以打

开该文件查看内容：



仅此而已！保存文件，甚至是大型文件，都可以用这种方法。你只需要记住你指定的文件名即可。另外，请注意，我上面的示例中，实际只存储了一个本地文件，不过你可以使用相同的代码来存储漫游文件，只不过使用 ApplicationData.Current.RoamingFolder 替换即可。

这里有个提醒:漫游文件并不会立即就传输的。至于什么时候传输，这是 Windows 控制的。

1.3. 总结

在 Windows 8 开发中设置和文件是很强大的一个工具。当在使用多台设备时，它很容易让你的程序变得很酷。

点击下面的图标，下载本文的示例代码：

明天，我将介绍动态磁贴(Live Tiles)，以及如何创建 primary tile 和 secondary tile，也会介绍如何更新他们。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