Windows 8 开发 31 日

第 14 日

地理位置

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 1 章　第 14 日地理位置

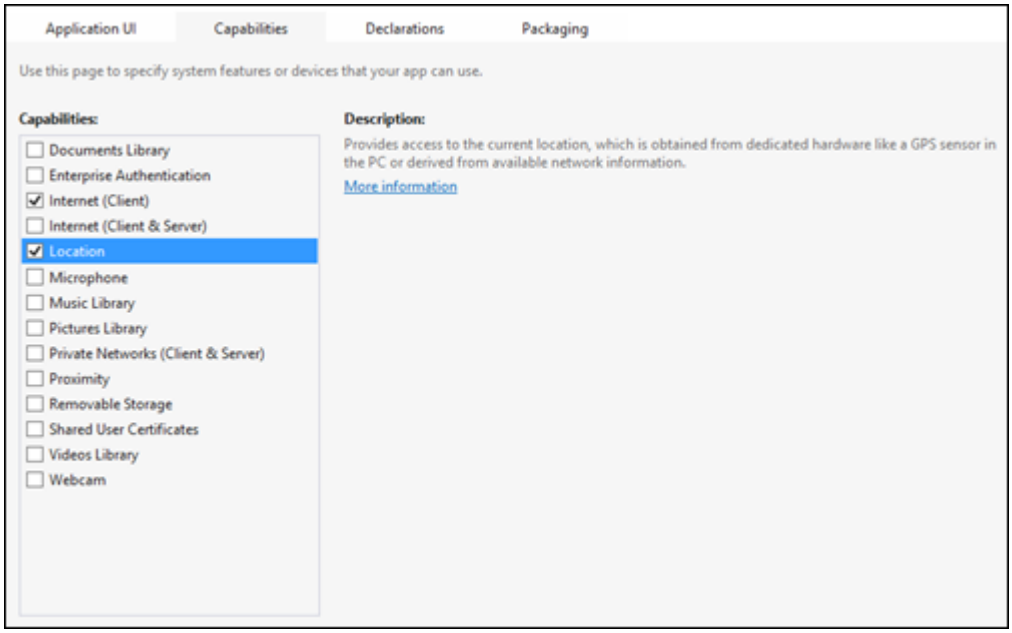
1.0. 介绍

今天，我来谈谈在任何一个平台上开发我都喜欢的一个特性：地理位置。知道用户的位置信息可以让每个程序更好。这些程序都有可能用到地位位置：游戏、地图、旅行和健身。

下面，我们就开始吧。

1.1. 更新 Manifest 文件

很不错的是，获取地理位置数据非常简单，首先，我们必须声明一个位置功能，才可以使用。打开 package.appxmanifest 文件的 Capabilities 选项，然后勾选 Location 框。



如果你忽略了这一步，而直接去写代码，你不会获得任何错误，但是你不会得到任何地理位置数据。

1.2. 获取地理位置数据

在本示例中，我将不断的跟踪设备的位置。首先让我有点迷糊的是在程序中，如何向用户请求开始跟踪设备位置的权限。如果你还记得，在第 11 日锁屏程序中，我们必须明确的提示用户授权访问锁屏。然后在 Loation 中，我们不需要这样做。当你试图访问位置数据的时候，这将会自动提示用户的。下面是相关代码。（我将所有的数据都设置给 MainPage.xaml 中的 TextBlocks，然后将其显示出来）

```
Geolocator location;

protectedoverridevoid OnNavigatedTo(NavigationEventArgs e)
```

```
{
        location = newGeolocator();
        location.PositionChanged += location_PositionChanged;
        location.StatusChanged += location_StatusChanged;
//GetLocationDataOnce();
}
```

```
protectedoverridevoid OnNavigatingFrom(NavigatingCancelEventArgs e)
{
        location.PositionChanged -= location_PositionChanged;
        location.StatusChanged -= location_StatusChanged;
}
```

```
asyncvoid location_PositionChanged(Geolocator sender, PositionChangedEventArgs args)
{
await Dispatcher.RunAsync(CoreDispatcherPriority.Normal, () =>
        {
Geoposition position = args.Position;

                LatitudeValue.Text = position.Coordinate.Latitude.ToString();
                LongitudeValue.Text = position.Coordinate.Longitude.ToString();
                AccuracyValue.Text = position.Coordinate.Accuracy.ToString();

                TimestampValue.Text = position.Coordinate.Timestamp.ToString();

if (position.Coordinate.Altitude != null)
                        AltitudeValue.Text = position.Coordinate.Altitude.ToString()
                                                + "(+- " + position.Coordinate.AltitudeAccuracy.ToString() + ")";
if (position.Coordinate.Heading != null)
                        HeadingValue.Text = position.Coordinate.Heading.ToString();
if (position.Coordinate.Speed != null)
                        SpeedValue.Text = position.Coordinate.Speed.ToString();
        });
}
```

```
asyncvoid location_StatusChanged(Geolocator sender, StatusChangedEventArgs args)
{
await Dispatcher.RunAsync(CoreDispatcherPriority.Normal, () =>
        {
                StatusValue.Text = args.Status.ToString();
        });
}
```

可能你已经猜测到了，这是一个事件驱动操作。当我的页面加载时，我创建了一个新的 Geolocator 对象，然后添加两个事件：PositionChanged 和

StatusChanged。

PositionChanged 是非常重要的一个。在这里面，我们获取有用数据的地方，比如 Longitude、Latitude 和 Accuracy。所有的这些值都来自 Geocoordinate 类，如上面代码所示，这里还有更多的数据：

- **精度(Accuracy)**是一个尺度，表示以米为单位，经纬度为中心点的一个圆。如下：

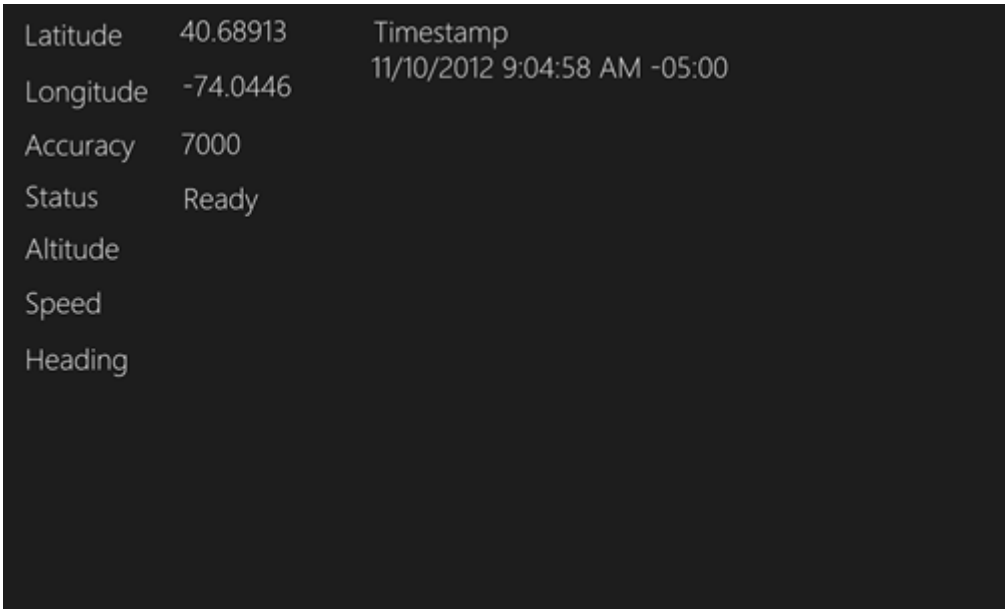
- **时间戳(Timestamp)**是读取位置信息的实际时间值。其中包括时区信息，所以你需要将这个时间转换为 UTC（国际协调时间——Universal Time Coordinated）。

- **海拔(Altitude)**是设备的海拔高度，以米为单位。这个值一般在晴雨表中用到(barometer)（在后面的文章中，会有提到），获得这个信息，也是是很重要的。
- **海拔精度(AltitudeAccuracy)**跟 Accuracy 一样，只不过是用在海拔的误差。
- **方向(Heading)**同样只在适当的场景使用，如罗盘。以相对于真北(true north)的度数为单位。
- **速度(Speed)**，从技术的角度上，可以通过跟踪维度/精度随时间的变化而计算出，不过我们开发者使用起来非常简单，可以直接获得这个速度值，以米/每秒为单位。不过这是一个可选值，只有在具有 GPS 传感器的设备才能获得。也就是说并不是所有的设备都能返回这个值。

上面示例中，另外的代码，我想说一下关于 async/await。你可能已经注意到，两个 event handler 函数都是以 async 开头，然后有如下代码：

```
await Dispatcher.RunAsync(CoreDispatcherPriority.Normal, () =>{};
```

因为获取位置数据是通过后台线程的，所以我们需要一个 Dispatcher 来回到主 UI 线程中——将数据显示出来。如果没有这个代码，那么直接操作 UI 层的话，会得到一个错误，因为后台线程不能直接访问 UI。下面是我的 UI 截图



现在我们可以获取到位置数据了，不过之前，我们还创建了另外一个 event handler：StatusChanged。它有 6 个不同的状态值：

- **未初始化(NotInitialized)**：当用户还未请求访问位置数据时会返回该值。
- **禁用(Disabled)**：当用户拒绝你访问位置信息是，会返回该值。
- **不可用(NotAvailable)**：当用户设备不支持位置数据时，返回该值。
- **没有数据(NoData)**：如果用户的设备支持位置，但是不能获得任何数据时返回该值。当用户没有连接 wi-fi，或者再室内，可能会发生这样的情况。
- **初始化中(Initializing)**：在用户授权获取位置数据后和真正开始接收到数据之间会返回该值。在你的程序中，当你第一次等到获取数据时同样会返回该值。
- **准备就绪(Ready)**：已经做好了响应准备，你可以获取到数据了。

本文我提供的示例程序，可以持续的获得用户位置的改变。如果你只需要获取一次位置数据，比如标记一下用户刚刚照的相片位置，那么你只需要请求 Geolocator 对象的 GetGeopositionAsync()方法即可，而不需要创建上面提到的两个事件。如下代码所示：

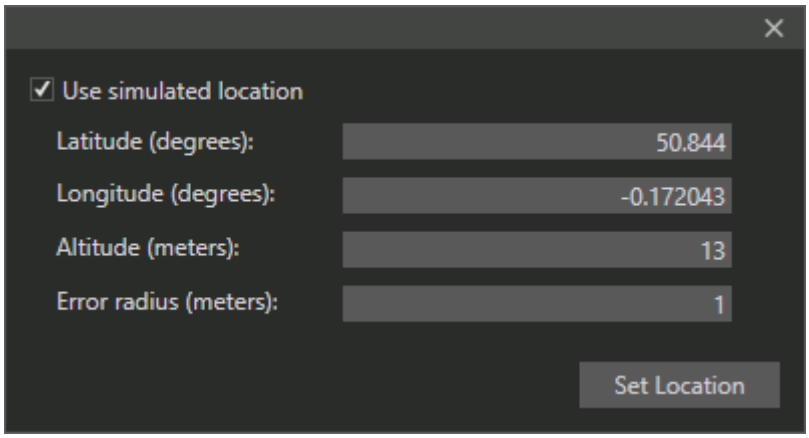
```
asyncprivatevoid GetLocationDataOnce()
{
Geoposition position = await location.GetGeopositionAsync();

      LatitudeValue.Text = position.Coordinate.Latitude.ToString();
      LongitudeValue.Text = position.Coordinate.Longitude.ToString();
      AccuracyValue.Text = position.Coordinate.Accuracy.ToString();
}
```

在上面，一次获得一个值。千万别创建一个循环来调用上面的方法，它非常的慢，通过上面提到的，注册相关事件，可以节约大量的系统资源。

1.3. 使用模拟器模拟数据

如果你使用传统的桌面 PC，那么你可能会发现，你的机器不能不能准确的判断出你的位置。虽然可以获取到网络位置（wifi），但是获取到的位置信息不太精确。通过模拟器，只需要打开 Location Settings，就可以设置一个精确的位置：

在这里，你可以输入纬度、经度、海拔以及精确度值：



就是这样啦！你现在可以写程序来判断用户在哪里了，并且不必到处走就可以模拟在地球上的任何位置，成本很低吧。

1.4. 总结

今天我介绍了地理位置数据，这对于商店里面的程序非常重要。

点击下面的图片，下载本文示例程序：

明天，我将介绍 on-screen keyboard(软键盘)、输入范围和文本框。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