Windows 8 开发 31 日

第 19 日

文件选择器

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 19 日文件选择器

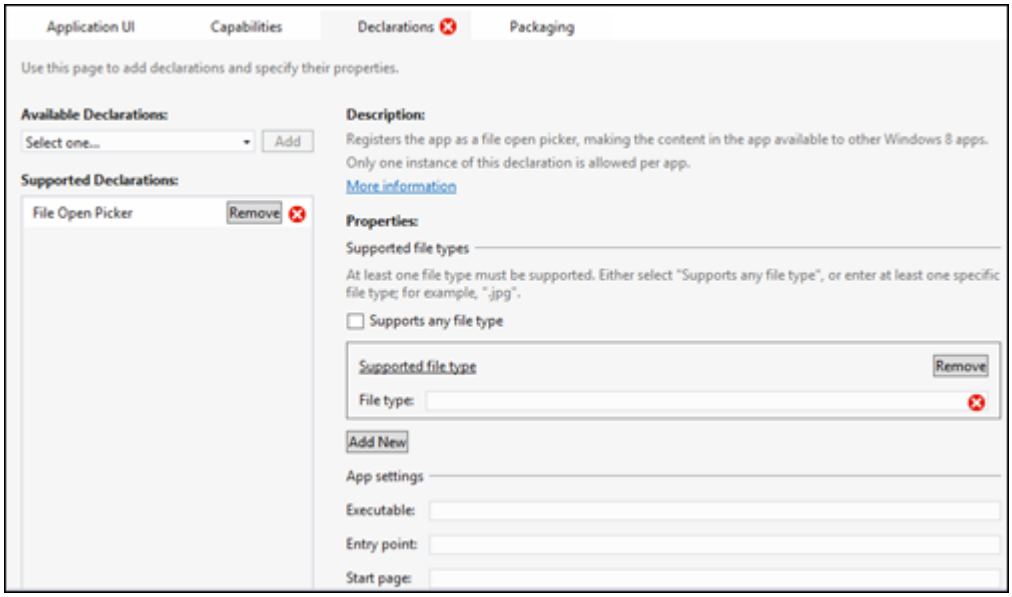
1.0. 介绍

今天，我将介绍 Windows 8 开发中另外一个非常好的工具：文件选择器。可以从用户机器上获取一个或者多个文件。最后我也会介绍一下如何在用户机器上选择一个文件夹。下面我们就开始吧。

1.1. 更新 package.appxmanifest

我们与用户系统进行交互的所有机制，首先都需要更新 package.appxmanifest 文件。在本文，我们要进行文件的打开和保存，所以所以需要添加相应的 declarations。

针对文件打开选择器，需要添加如下内容：

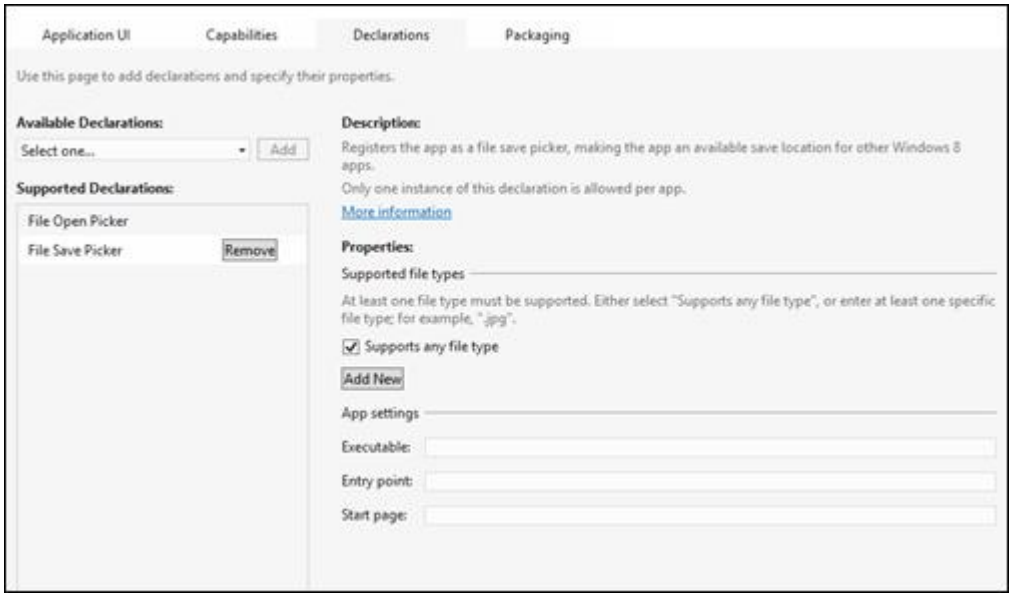


如上图，有两个选择：

Supports any file type。选择这个的话，你的程序就被声明为可以访问任意类型的文件。

Supported file type。还尅在这里指定你想要打开的文件类型，例如你只想让程序打开 Excel 文件，那么可以在这里输入“.xls”。

在我的示例中，我选择“**Supports any file type**”，文件保存选择器的设置也是类似的。下面是我选择的截图：



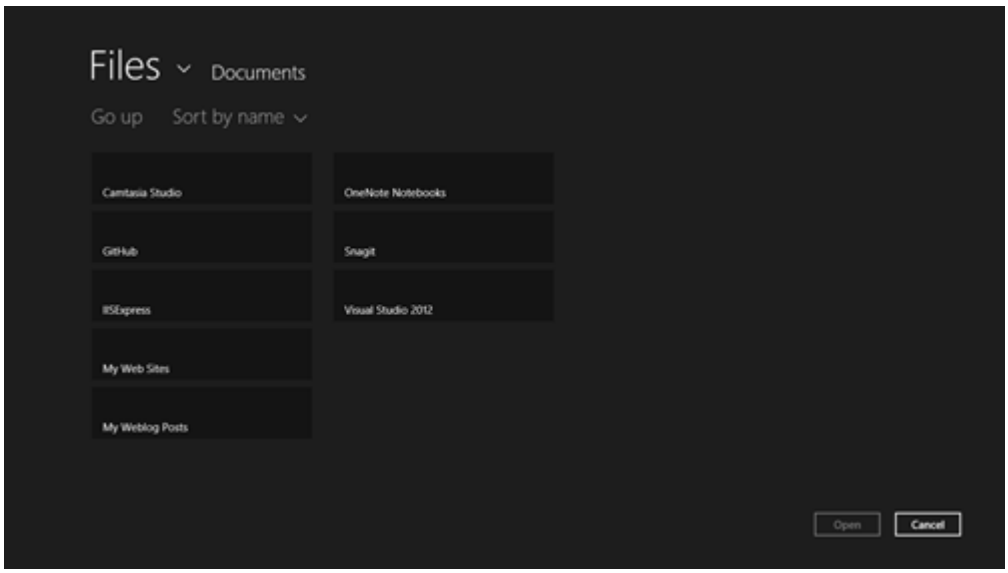
1.2. 从用户机器上获取一个文件

本文创建从用户机器上选择文件的示例代码越来越复杂。首先是选择一个文件，然后是多个文件，接着是选择指定某种文件类型（如.png）的多个文件。由于大多数代码都是相同的，所以，将主要集中在每一步新增的代码。

从用户机器上选择一个文件，需要从 FileOpenPicker 对象开始。这个对象会给用户打开一个文件打开选择器对话框，帮助用户选择一个或者多个文件，然后返回给我们。它有许多选项，我这里会进行介绍。下面是选择指定文件类型的一个文件的相关代码：

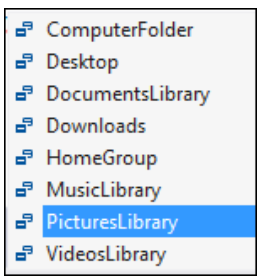
```
FileFileOpenPicker picker = newFileOpenPicker();
picker.FileTypeFilter.Add(".xls");
StorageFile file = awaitpicker.PickSingleFileAsync();
```

在上面的代码中，创建 FileOpenPicker 时，我们必须设置 FileTypeFilter。不允许提供一个通配符或者忽略该过滤器。你必须提供至少一个文件类型，也可以提供多个（比如在选择图片是，可能会有 PNG、JPG、JPEG、GIF 和 BMP 等），运行上面的代码，用户会看到类似如下的界面：



这就是文件打开选择器的界面，当用户选择了一个文件后，它将给我们的程序返回一个 StorageFile，StorageFile 在之前的学习中，我们已经多次使用了。

下一步，我们可能有更好的一个主意：打开文件打开选择器的时候，预先指定一个路径，比如图片集合、或音乐。我们可以指定如下用户电脑中 8 个位置里的一个：

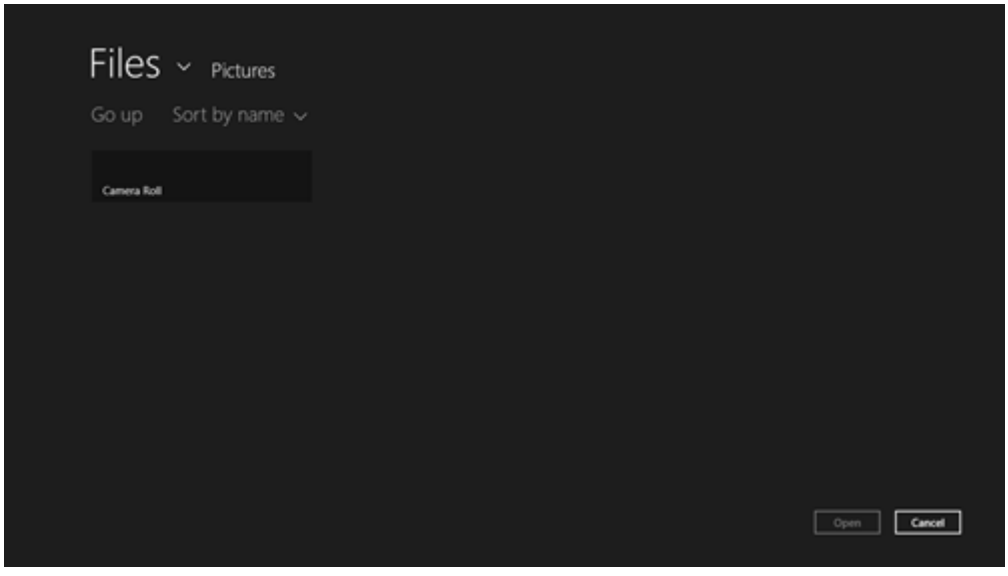


Ok，要想让文件打开选择器打开上面几个位置中的一个时，需要再方法中添加一行代码，如下：(我另外还添加了一些新的文件过滤类型)

So, to get the File Picker to open in one of these locations, you need to add one line of code to your method, so that it looks like this (you can see I've added a few new filters as well):

```
FileOpenPicker picker = newFileOpenPicker();
picker.FileTypeFilter.Add(".png");
picker.FileTypeFilter.Add(".jpg");
picker.FileTypeFilter.Add(".gif");
picker.FileTypeFilter.Add(".bmp");
picker.SuggestedStartLocation = PickerLocationId.PicturesLibrary;
picker.ViewMode = PickerViewMode.List;
StorageFile file = awaitpicker.PickSingleFileAsync();
```

这样，文件打开选择器就会直接启动到指定的位置（在这里，是图片库）：

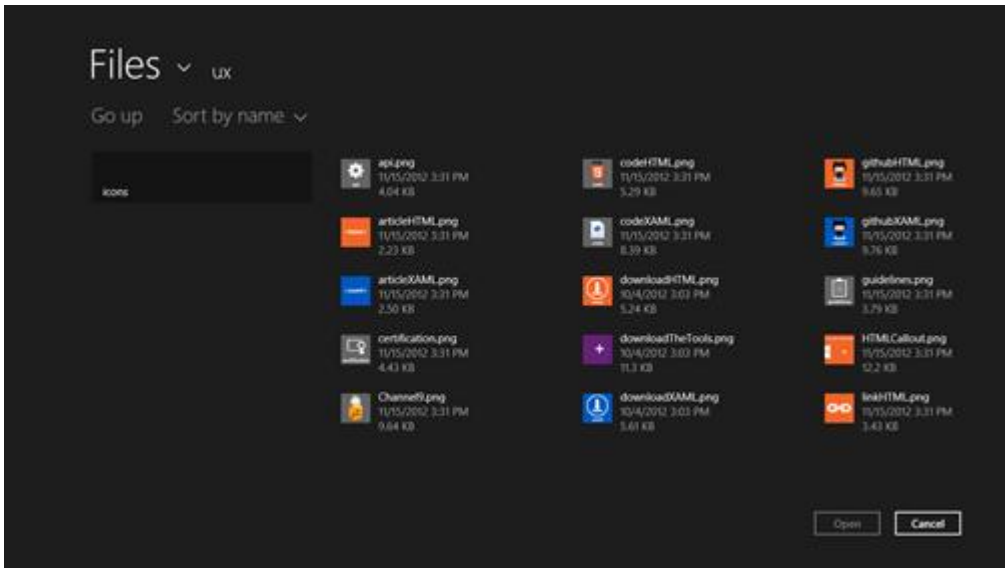


最后，我们还可以设置文件打开选择器中如何显示文件。使用 FileOpenPicker 对象的 ViewMode 属性即可，如下：

```
picker.ViewMode = PickerViewMode.List;
```

可选项只有 List 或 Thumbnail，它们显示出来的样子不太一样。Thumbnail 只显示一个方形图片，代表一个文件。List 也显示一个 icon，不过还显示与文件相关的信息。看来分别如下（点击看大图）：

List



Thumbnail

1.3. 从用户机器上获取多个文件

有时候，我们希望用户一次可以选择多个文件。这种情况的话，我们调用 FileOpenPicker 对想不同方法即可。这里调用 PickMultipleFilesAsync()方法。

```
FileOpenPicker picker = newFileOpenPicker();
picker.FileTypeFilter.Add(".png");
picker.FileTypeFilter.Add(".jpg");
picker.FileTypeFilter.Add(".gif");
picker.FileTypeFilter.Add(".bmp");
picker.SuggestedStartLocation = PickerLocationId.PicturesLibrary;
picker.ViewMode = PickerViewMode.Thumbnail;
IReadOnlyList<StorageFile> files = awaitpicker.PickMultipleFilesAsync();
```

从代码来看，选择一个和多个文件，或多或少有点不同。我们在这里收到的不是一个 StorageFile，而是一个 StorageFile 集合，集合中是用户选择的多个文件。

如下图，是可以选择多个文件的文件打开选择器。

从上图中可以看到，不仅可以选择多个文件，还可以在文件打开选择器界面底部看到用户选中的文件列表。它还支持一次从不同文件夹中选择多个文件。也就是说用户可以在一个文件中选择几个文件，然后在另外一个文件中选择几个文件，这些被选中的文件，可以一次返回给我们的程序，而不用用户多次操作。这个功能应该是非常棒的，请别忽视它。

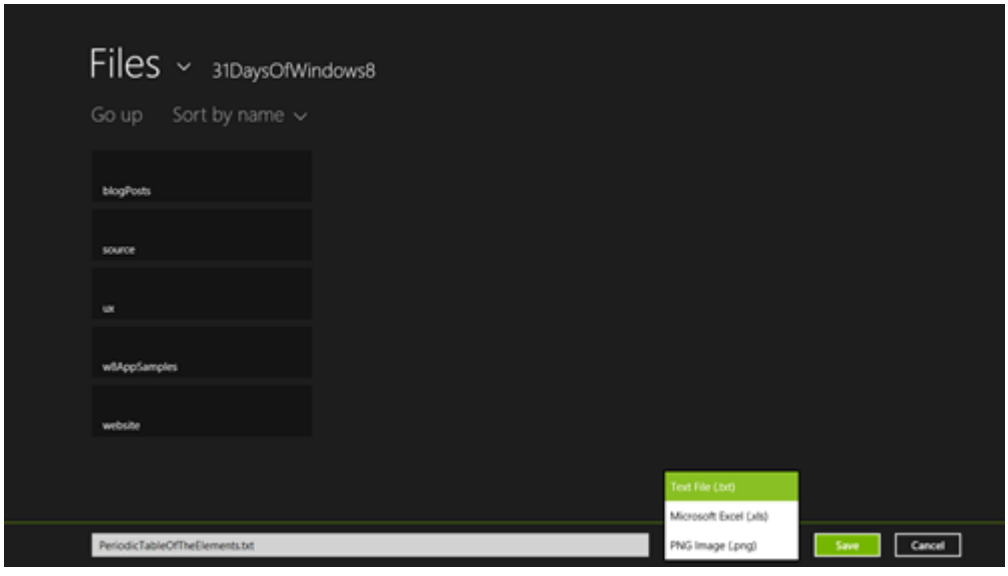
接下来，我们看看如何使用文件选择器，将文件保存到用户机器中。

1.4. 将文件保存到用户的机器上

与上面操作相关的一个功能，可能就是我们希望能够将一个文件保存到用户机器中。这与本系列中第 8 日——将文件保存在程序内部是不同的。在这里，我们是将文件永久的保存在用户机器中，即使我们的程序被卸载了，保存的文件还存在。如果别的程序不能使用你的文件，这可能不是你想要的一种存储方法。我

建议这种方式存储的文件一般可以被许多其它程序打开。要实现这样的文件保存功能，与 FileOpenPicker 非常类似，只不过使用的是 FileSavePicker。

另外一个不同点就是我们将数据写到用户的硬盘中。这也就会带来一些其它问题，比如文件已经被打开，并被其它程序编辑，甚至删除该文件。所以，我们首先需要启动文件保存选择器，让用户决定将文件保存在什么地方（同时还有文件类型和文件名），看起来如下：



下面的代码是文件保存的一个完整处理过程：

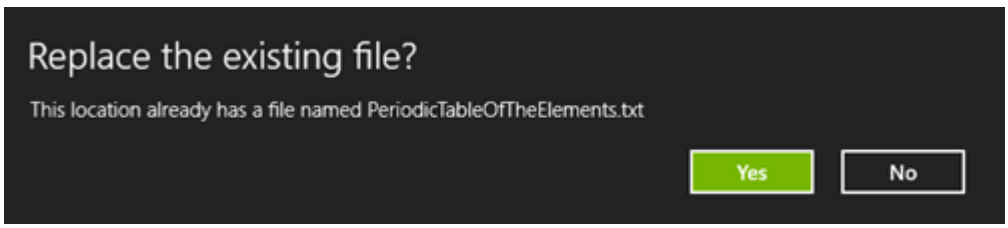
```
FileSavePicker saver = newFileSavePicker();
saver.SuggestedStartLocation = PickerLocationId.Desktop;
saver.FileTypeChoices.Add("Text File", newList<string>() { ".txt" });
saver.FileTypeChoices.Add("Microsoft Excel", newList<string>() { ".xls" });
saver.FileTypeChoices.Add("Image", newList<string>() { ".png", ".jpg", ".bmp" });
saver.SuggestedFileName = "PeriodicTableOfTheElements";
StorageFile file = awaitsaver.PickSaveFileAsync();

if (file != null)
{
CachedFileManager.DeferUpdates(file);
awaitFileIO.WriteTextAsync(file, "This is a link to the Periodic Table of the Elements.    http://www.ptable.com/
You didn't expect to find all of the contents here, did you?");
FileUpdateStatus status = awaitCachedFileManager.CompleteUpdatesAsync(file);
```

```
}
```

在上面代码中，开始的地方看起来就有点与之前的不同了。不再使用过滤器，而是添加相应的 FileTypeChoices，代表用户可以存储的不同文件格式。我选择的这些 choice 有点不太相关，不过这可以帮助演示在列表中显示指定的任意文件类型。这些都是由你来控制的。

另外，当用户选择了一个位置和对应文件的 name 后，还没有完。实际上，选择器会将这些值返回给我们，下一步就是利用这些值将数据写入到对应的文件中。在我的 if 语句中，我使用 DeferUpdates()方法来保护正在使用的这个文件不别的程序操作，直到我的操作完成。下一步是将 content 写到文件中。最后，使用 CompleteUpdateAsync()方法来提交这些保存的数据，这就意味着所有的工作都完成。

如果文件已经存在，会有如下的提示“Replace the existing file”对话框。



Ok，上面就是保存一个简单文件到用户系统中的过程。更复杂的操作和文件类型，请研究 FileIO 类，可以在 msdn 上看到更多相关内容。

1.5. 在用户机器上选择一个文件夹

最后我还想介绍与本文相关的另外一个主题：从用户机器上选择一个文件夹，而不是一个指定的文件。让用户来选择一个默认的存储位置，或者选择一个你查找文件的位置。这种处理的一个好处就是当用户选择一个文件夹后，我们可以将该文件夹保存为我们的默认文件夹，并且获得确定的授权：在不需要进行请求访问下，对该文件夹进行读写。下面是相关代码：

```
FolderPicker picker = newFolderPicker();
picker.FileTypeFilter.Add(".xls");
StorageFolder folder = awaitpicker.PickSingleFolderAsync();
if (folder != null)
{
StorageApplicationPermissions.FutureAccessList.AddOrReplace("DefaultFolder", folder);
}
```

如上所示，前三行代码似曾相识。如果有一个实际的文件夹返回来了，那么我们可以将这个文件夹保存到 StorageApplicationPermissions.FutureAccessList 中，即我们已经授权可以使用了，我们之后就可以轻松的访问存在这个 FutureAccessList 中的内容了。

1.6. 总结

今天，我介绍几种用户与其机器上的文件交互的方法。我们学习了如何打开单个或者多个文件，将文件保存到硬盘中，并且学习了如何选择一个默认的保存路径，以供之后使用。你会发现这些方法在你的程序中会经常被使用到，所以，这是很重要的一课。

点击下面图片，可以下载到本文的示例代码：

明天，我将介绍另外一个常用功能：打印。我们会看到如何与用户的打印机通信，以及如何将程序与系统注册，使打印更简单。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