Windows 8 开发 31 日

第 07 日

共享合约

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 07 日共享合约

1.0. 介绍

过去两天，我们都在学习 Windows 8 的新特征：合约。通过将设置合约融合到程序中介绍了合约的概念。然后学习了如何将搜索扩展我们的程序供用户使用。今天我将介绍共享合约.

Windows 8 之前的系统,想要在程序中构建“社交”是很困难的。你不仅要学习程序运行平台的 APIs，还得学习其它一些 APIs，如 Facebook，Twitter，以及其它你想加入的社交网络。

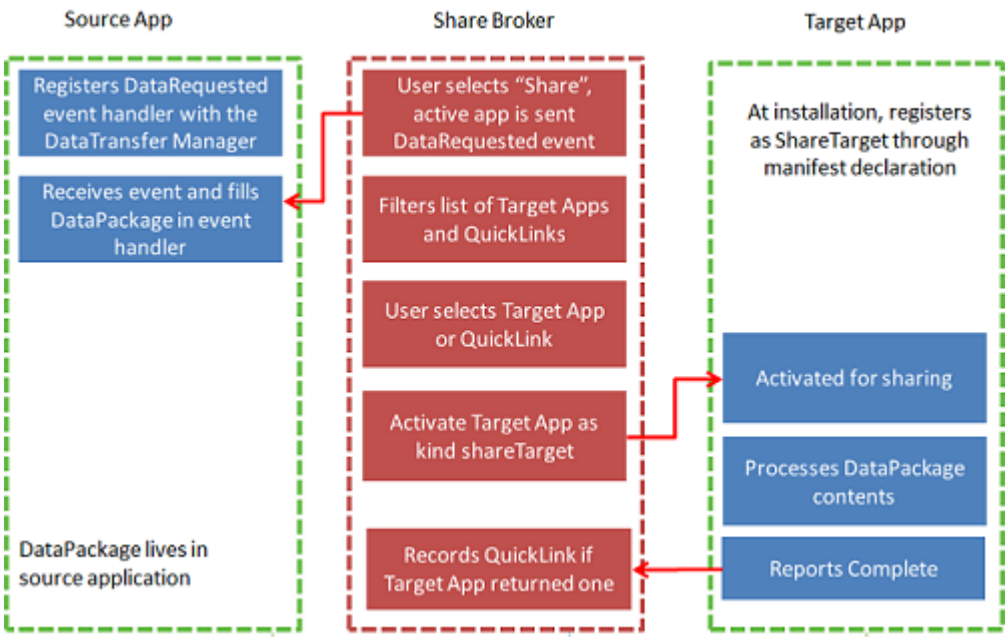
这个工作任务是很繁重的,要想做到高效率是不可能的。

而在 Windows 8 中，我们只需要关注我们创建的程序即可。实际上,在 Windows 8 程序中放一个按钮(比如:分享到社交网 X 上)是不符合 Windows 8 的设计指南的。当用户没有使用 Twitter 的时候，为什么要提供一个 Twitter 按钮呢？其它的比如 Google+ ,Facebook，Flickr，GitHub 或者其它社交平台也一样。

使用共享合约,用户可以完全控制共享.你只需要准备好需要共享的内容,然后由用户决定共享到哪里,以及如何共享.这样会给用户很好的体验,并且这也是 Windows 8 中非常棒的一个亮点.

1.1. 共享的定义

共享涉及到两方面,谁共享(source app)和谁接收(target app)。其中，是通过 broker 代理的。如下图所示.



图片来自:

http://msdn.microsoft.com/en-us/library/windows/apps/hh758314.aspx

你能共享什么类型的文件呢?这里有 6 种不同的文件类型能够共享:

- 无格式的纯文本(Unformatted Plain Text)
- 链接(Link)
- 带格式的 Content/html(Formatted Content / HTML)
- 文件(Files)
- 单个图片(Single Image)
- 自定义格式的数据(Custom Data Format)

数据的共享是通过一个叫做 DataPackage 的对象。在本文中,我将共享源和共享目标分为两节进行讲解。

1.2. 共享源

在这里,我将介绍如何让上面提到的 7 种类型可以被共享。首先,需要创建一个新的 Blank App Template。在本文的示例代码中,你会注意到,我为每种类型的数据创建了一个单独的页面,当然,这不是必须的。不过请记住,一次只能共享一种类型的数据。下面就从无格式的纯文本开始吧。

1.2.1. 共享无格式的纯文本

在这里的示例,将会是共享源中最长的示例,因为每个示例的代码大多数都是相同的。所以我只在这个示例中介绍所有内容,而在后面的示例中我将集中在不

同的地方进行介绍。

首先，我们需要引用到 DataTransferManager。同时需要将 Windows.ApplicationModel.DataTransfer 名称空间添加到页面中。在下面的代码中，你会看到我实例化了一个 DataTransferManager。并创建了一个 event handler(在 OnNavigatedTo()方法中)，当该页面发生了共享请求时,该 event handler 会被触发。

```
DataTransferManager dtm;
```

```
protectedoverridevoid OnNavigatedTo(NavigationEventArgs e)
{
base.OnNavigatedTo(e);
      dtm = DataTransferManager.GetForCurrentView();
      dtm.DataRequested += dtm_DataRequested;
}
```

```
protectedoverridevoid OnNavigatingFrom(NavigatingCancelEventArgs e)
{
base.OnNavigatingFrom(e);
      dtm.DataRequested -= dtm_DataRequested;
}
```

上面的代码中，可以看到当离开这个页面时，我在 OnNavigatingFrom 方法中移除了 DataRequested 事件。在离开页面时,确保取消注册的 event handler 是一个很好的编程习惯。

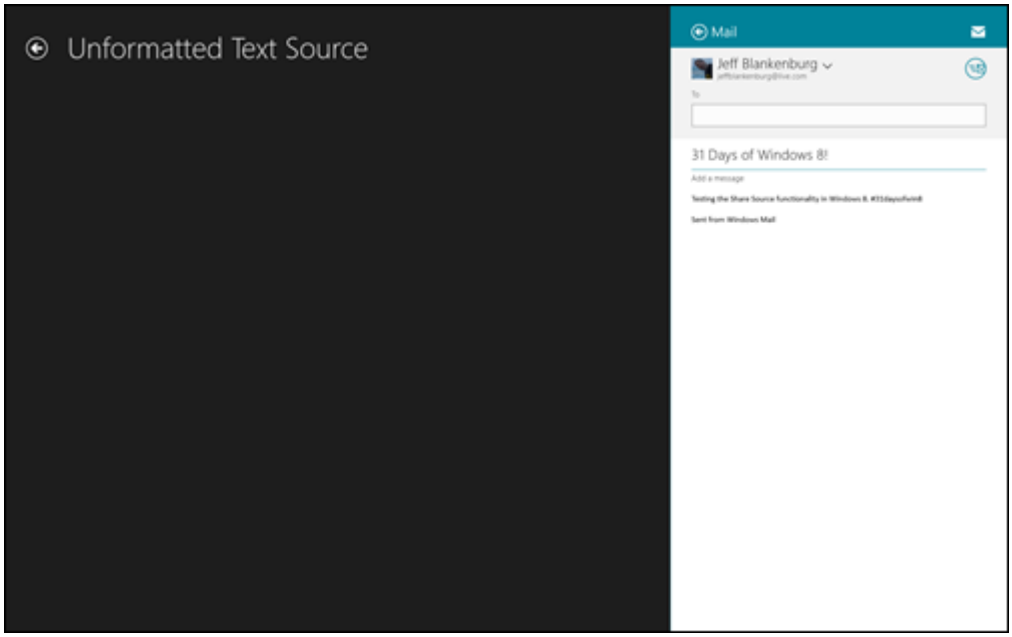
最后，是需要真正的共享一些文字信息。这是通过 dtm_DataRequested()方法共享的。

```
void dtm_DataRequested(DataTransferManager sender, DataRequestedEventArgs args)
{
string textSource = "Testing the Share Source functionality in Windows 8. #31daysofwin8";
string textTitle = "31 Days of Windows 8!";
string textDescription = "This just explains what we're sharing.";    //This is an optional value.
```

```
DataPackage data = args.Request.Data;
        data.Properties.Title = textTitle;
        data.Properties.Description = textDescription;
        data.SetText(textSource);
}
```

看上面的代码,我创建了一个 DataPackage 对象,然后设置它的属性。使用到的值我是写在代码中的，用户可以在下图中看到提示信息（点击看大图）:

一旦用户选择了一个程序(这里我选择 Mail 程序),将会看到填充到选中程序的数据。（点击看到图）:



剩下的数据类型,我将显示 dtm_DataRequested()方法中的内容，而其它的所有内容都基本保持不变。

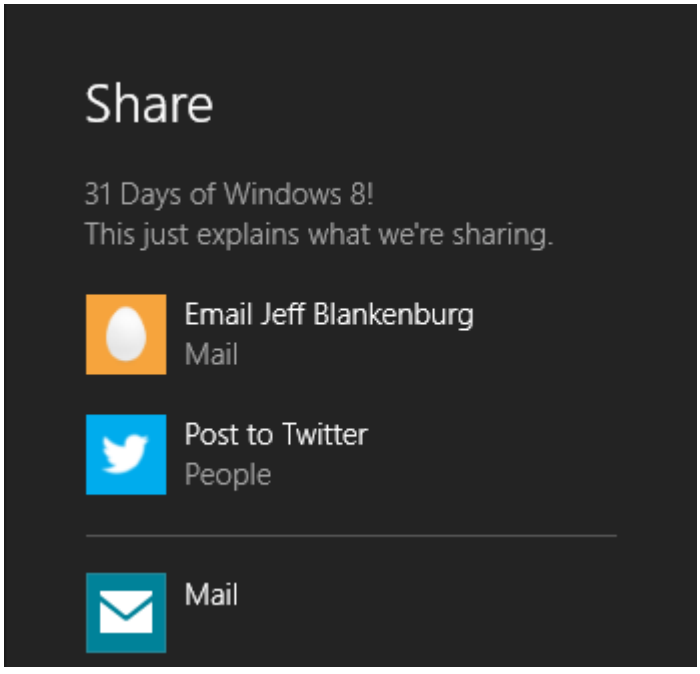
1.2.2. 共享链接

共享链接与纯文本唯一的不同就是你必须给 DataPackage 指定一个有效的 Uri 对象。下面是代码：

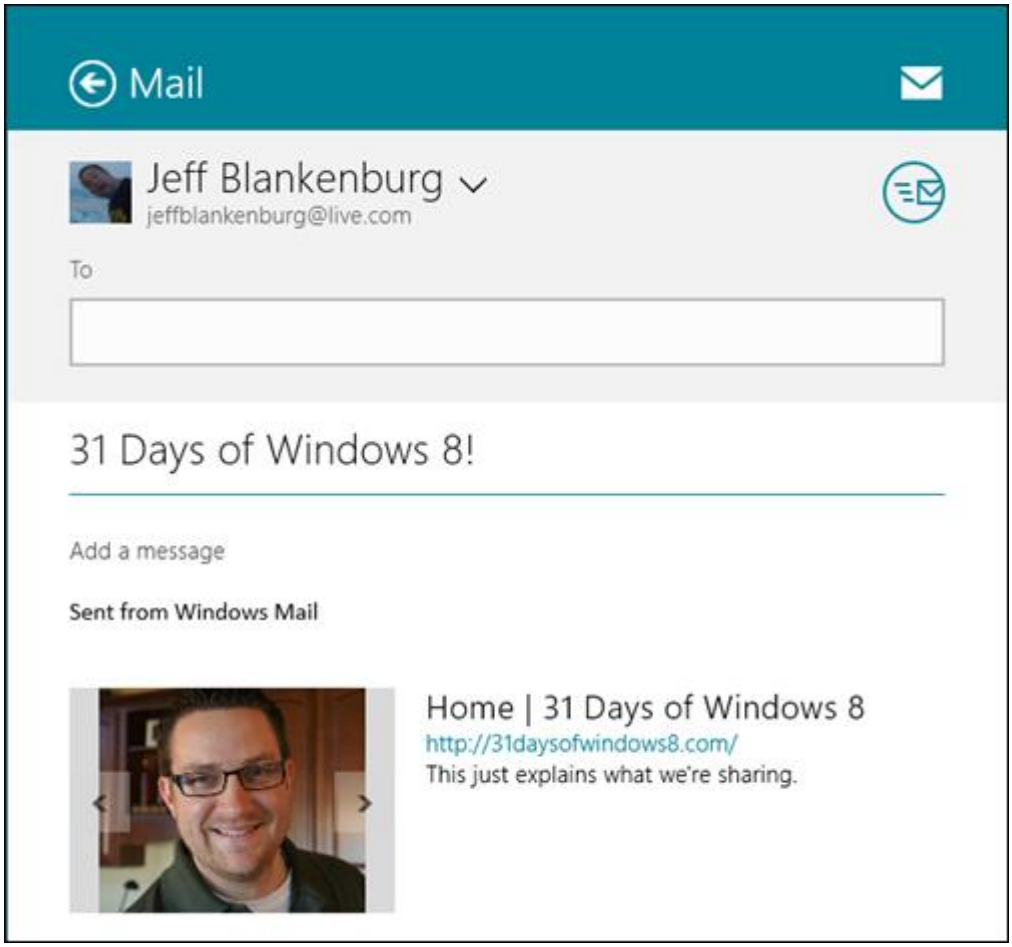
```
void dtm_DataRequested(DataTransferManager sender, DataRequestedEventArgs args)
{
Uri linkSource = newUri("http://31daysofwindows8.com");
string linkTitle = "31 Days of Windows 8!";
string linkDescription = "This just explains what we're sharing.";    //This is an optional value.

DataPackage data = args.Request.Data;
      data.Properties.Title = linkTitle;
```

```
        data.Properties.Description = linkDescription;
        data.SetUri(linkSource);
}
```

真正的不同地方是在显示出来的 UI.下面是最初的提示内容:



上图中,你会看到线的上面提示的是 People 程序,并建议我发表到 Twitter 上.如果我打开 People 程序后,我依然可以选择 Facebook。下面是打开 mail 程序后显示的数据：



在这里实现的一些功能非常的 cool:

Mail 程序自动去网页上抓取一些它认为有用的缩略图。所以我必须在这里说明一下为什么会有一个默认的图片在此出现。

Mail 程序同样会去抓取网站的“标题”，并将其显示出来。这非常实用。
在纯文本示例中，Mail 程序没有使用 Description 属性，而在这里，它被嵌入进去了。

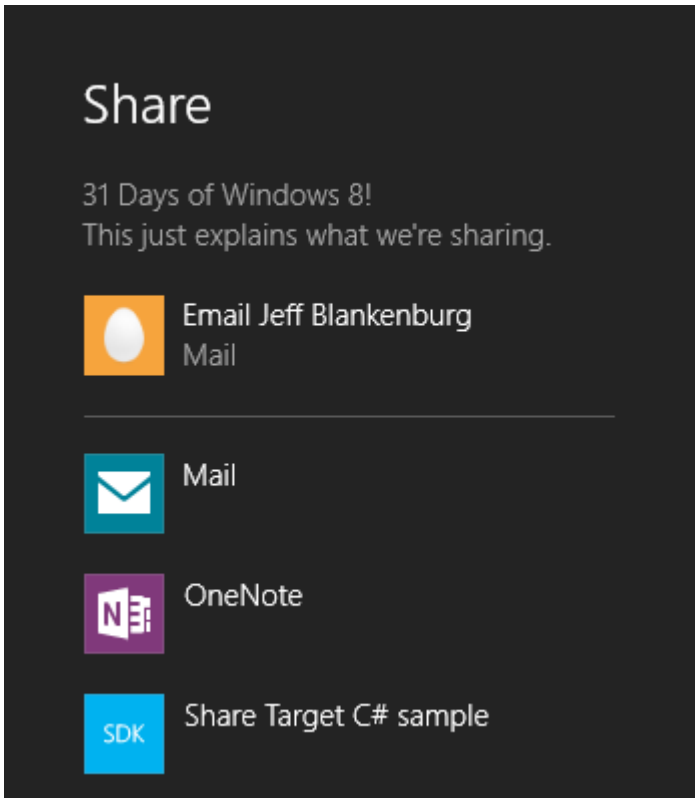
下面，我们来学习一下如何共享带格式的 Content/html。

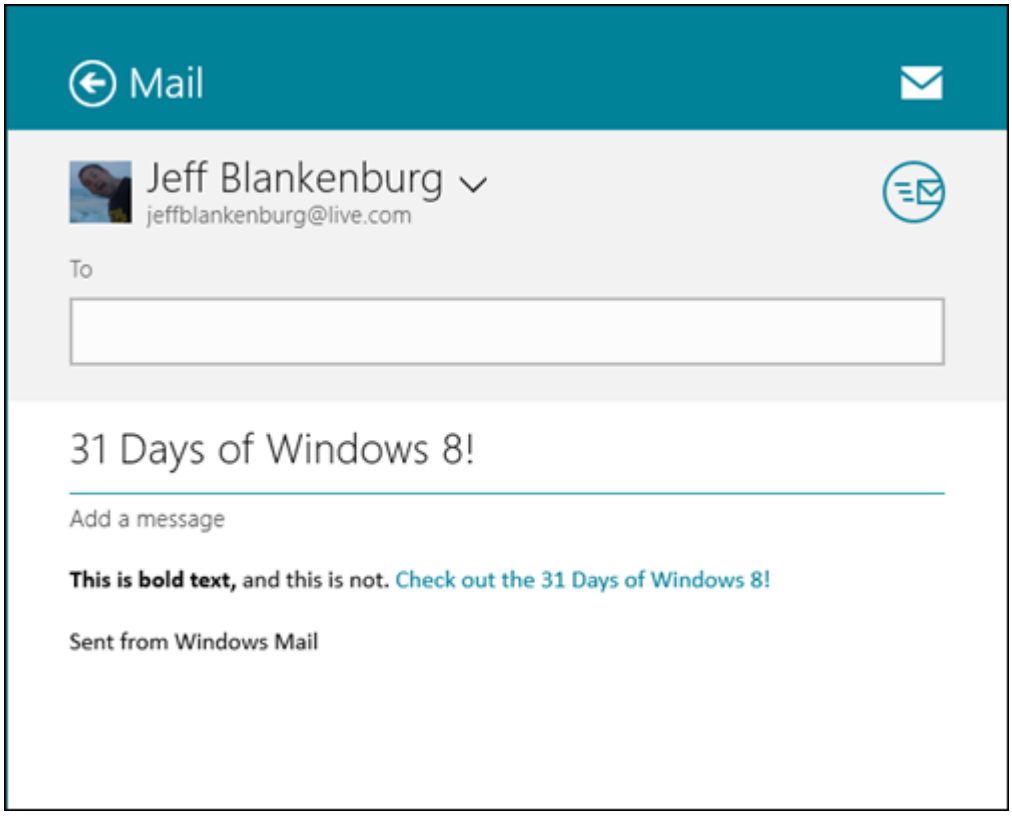
1.2.3. 共享带格式的 Content / HTML

下面是我实现的代码：

```
void dtm_DataRequested(DataTransferManager sender, DataRequestedEventArgs args)
{
string HTMLSource = "<strong>This is bold text,</strong> and this is not.    <a
href='http://31daysofwindows8.com'>Check out the 31 Days of Windows 8!</a>";
string HTMLTitle = "31 Days of Windows 8!";
string HTMLDescription = "This just explains what we're sharing.";    //This is an optional value.

DataPackage data = args.Request.Data;
      data.Properties.Title = HTMLTitle;
      data.Properties.Description = HTMLDescription;
      data.SetHtmlFormat(HtmlFormatHelper.CreateHtmlFormat(HTMLSource));
}
```

这里需要用不同的方式处理 content。上面的代码中，当我给 DataPackage 设置 content 时，我首先调用了 HtmlFormatHelper。HtmlFormatHelper 会使用另外一个原始的字符串，并将其转换为可用——html 标记能够被解析和使用。在共享面板出来时,这跟最初的共享提示看起来是一样的.但是当你选择 mail 程序时,你会看到格式化的 html 显示在 body 中。


Share
31 Days of Windows 8!
This just explains what we're sharing.
Email Jeff Blankenburg
Mail
Mail
OneNote
SDK
Share Target C# sample



这个程序的强大之处就在于可以创建 HTML 格式的 email 消息。如果不懂 html 的人来创建这些消息是非常困难的，想一下，如果一个程序可以让用户创建他们自己的 html 界面，然后将其共享到邮件客户端并发送出去，这是不是很棒。

1.2.4. 共享 Files

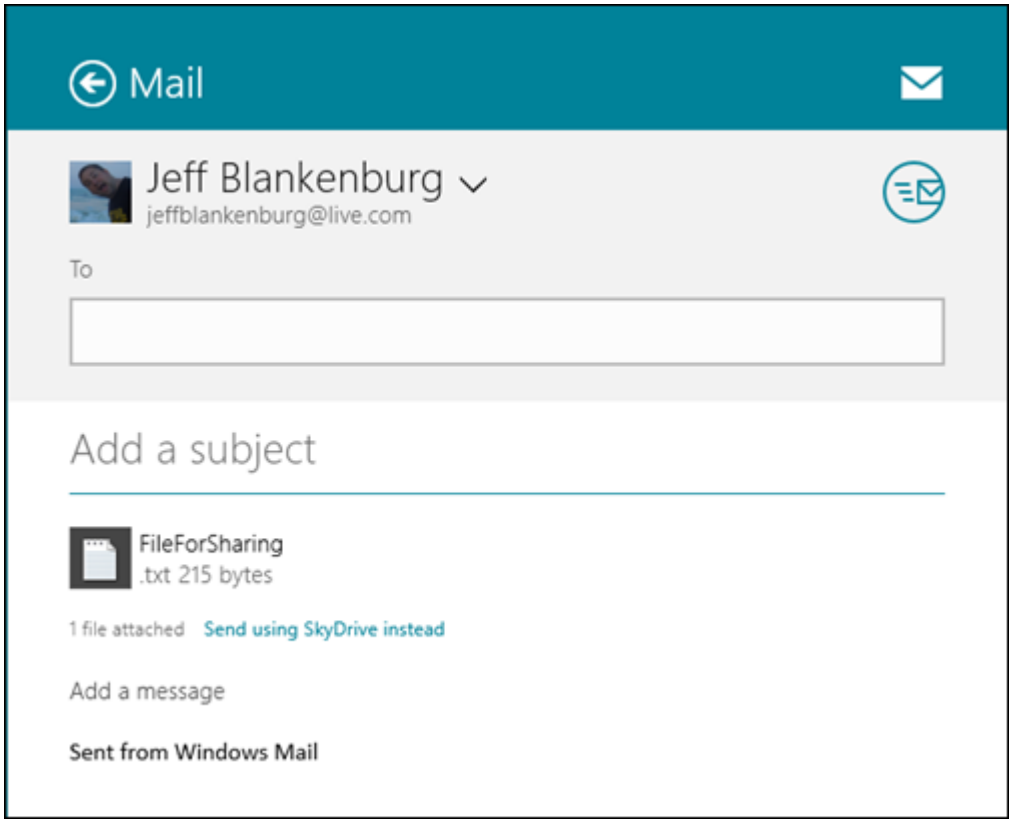
一个或者多个文件可以被共享。由于我们需要从 storage 中读取出文件,所以这里还需要写一点额外的代码。如果你已经有了一些需要共享的文件,那么只需要简单的通过异步方式获取到这些文件,然后将文件添加到 List<IStorageItem>集合中。下面我将通过异步的方式获取到文件。当获取完毕之后，开始共享文件。

```
asyncvoid dtm_DataRequested(DataTransferManager sender, DataRequestedEventArgs args)
```

```
{
string FileTitle = "31 Days of Windows 8!";
string FileDescription = "This just explains what we're sharing.";    //This is an optional value.

DataPackage data = args.Request.Data;
        data.Properties.Title = FileTitle;
        data.Properties.Description = FileDescription;

DataRequestDeferral waiter = args.Request.GetDeferral();

try
        {
StorageFile textFile = awaitPackage.Current.InstalledLocation.GetFileAsync("FileForSharing.txt");
List<IStorageItem> files = newList<IStorageItem>();
                files.Add(textFile);
                data.SetStorageItems(files);
        }
finally
        {
                waiter.Complete();
        }
}
```

注意，上面这个方法需要使用 async 关键字来修饰，因为该方法调用了异步方法。你将会在整个 Windows 8 开发过程中看到这种异步调用方式。它使得编码变得很容易。

依然，共享的提示跟之前的示例类似，当时当打开 mail 程序时，开起来是这样的：



有趣的是，之前的的共享中，mail 程序将属性 Title 作为邮件的主题，而在这个示例中，不是这样的，我相信微软的 product team 有一堆为什么这么做的理由，但是我认为，在这里最好统一起来。

1.2.5. 共享 Images

当共享一个图片时，你可能有两种选择：跟普通文件一样共享（上一节中所讲），另外一种则是跟普通文件不一样的共享方式。在这里的示例中，我将展示如何以两种不同的方式进行图片共享,那样目标程序就可以使用它们感兴趣的数据。程序寻找文件可以用，同样寻找图片也可以用。当你需要有多种类型的数据需要共享时,你可以使用这这里的小技巧,并且不只是同样的内容可以被以两种不同的

方式共享。

```
asyncvoid dtm_DataRequested(DataTransferManager sender, DataRequestedEventArgs args)
{
string FileTitle = "31 Days of Windows 8!";
string FileDescription = "This just explains what we're sharing.";    //This is an optional value.

DataPackage data = args.Request.Data;
      data.Properties.Title = FileTitle;
      data.Properties.Description = FileDescription;

DataRequestDeferral waiter = args.Request.GetDeferral();

try
      {
StorageFile image = awaitPackage.Current.InstalledLocation.GetFileAsync("Assets\\0.png");
            data.Properties.Thumbnail = RandomAccessStreamReference.CreateFromFile(image);
            data.SetBitmap(RandomAccessStreamReference.CreateFromFile(image));

List<IStorageItem> files = newList<IStorageItem>();
            files.Add(image);
            data.SetStorageItems(files);
      }
finally
      {
            waiter.Complete();
      }
}
```

上面代码可以看到,我调用了两个方法：SetBitmap 和 SetStorageItems。这两种方式，在共享提示中看起来是一样的，但是在每种情况下可用的程序列表是不同的。例如:Mail 程序不接受原始图像(raw image)，但是接受放在 StorageItems 中的图片。试一下上面的代码就知道了。令我感到奇怪的是 People 并不接受这两种类型的共享。在本文的后面点，我将介绍共享目标，其中就涉及到如何接收不同类型的数据。

1.2.6. 共享自定义格式的数据

可能你需要多次共享自定义的数据。99%的情况下,这些自定义数据符合标准的模式,比如在这个可以发现的 http://schema.org/。可能你不希望使用预先定义好的模式,但是请确认你理解了:guidance Microsoft offers for creating your own custom data formats。

简单的说,共享自定义数据,你需要制定一个 DataPackageFormat ,虽然可以是任何你想要的数据格式。在大多数情况下,共享自定义格式的数据是用在你自己的两个程序中,不过使用标准的模式,可以增加你共享的数据可以被别的程序使用的几率。最基本的,你会指定一个字符串代表你的数据模式,如果其它作为目标的程序,同样指定了相同的字符串,那么在共享程序列表中,会显示出该目标程序.

```
void dtm_DataRequested(DataTransferManager sender, DataRequestedEventArgs args)
{
string customData = @"{
            ""type"" : ""http://schema.org/Person"",
            ""properties"" :
            {
            ""image"" : ""http://jeffblankenburg.com/images/200x200.png"",
            ""name"" : ""Jeff Blankenburg"",
            ""affiliation"" : ""Microsoft"",
            ""birthDate"" : ""07/22/1976"",
            ""jobTitle"" : ""Senior Technical Evangelist"",
            ""nationality"" : ""United States of America"",
            ""gender"" : ""Male""
            }
      }";
string linkTitle = "31 Days of Windows 8!";
string linkDescription = "This just explains what we're sharing.";    //This is an optional value.

DataPackage data = args.Request.Data;
      data.Properties.Title = linkTitle;
      data.Properties.Description = linkDescription;
      data.SetData("http://schema.org/Person", customData);
}
```

在上面的示例中，我使用了的模式来自：http://schema.org/Person，而目前在我的机器上没有别的程序可以识别这种模式，所以我会得到如下的共享提示：

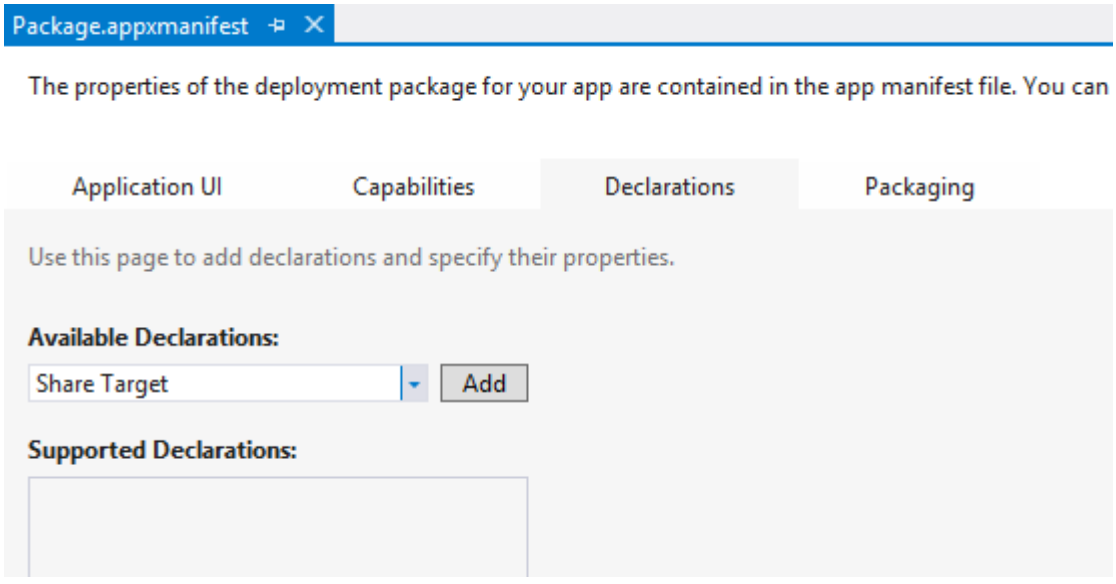
依然如上面所有示例中你所看到的,Title 和 Description 继续显示在相同的地方.但是,这次并没有显示可用于共享的程序.在下一节中,我将介绍如何使程序能够接收这样的数据类型.

1.3. 共享目标

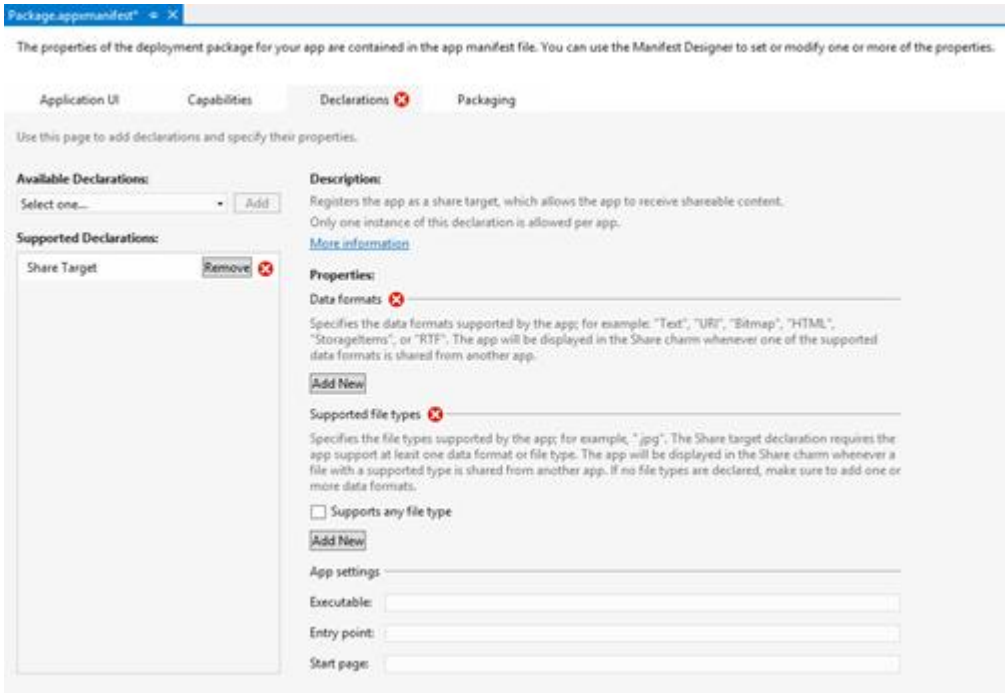
当我刚开始写本文的时候，我计划将共享目标放到最后，因为当时我认为共享目标要更难一点（还有更长）。而事实证明，我错了。共享目标大部分的工作是在 Package.appmanifest 文件中。

打开 Package.appmanifest 文件，定位到 Declarations 选项，你会发现这里几乎

一片空白。我们需要做的是通过下拉列表中选择 Share Target，并单击 Add 按钮来添加共享合约(Share Target)的声明。

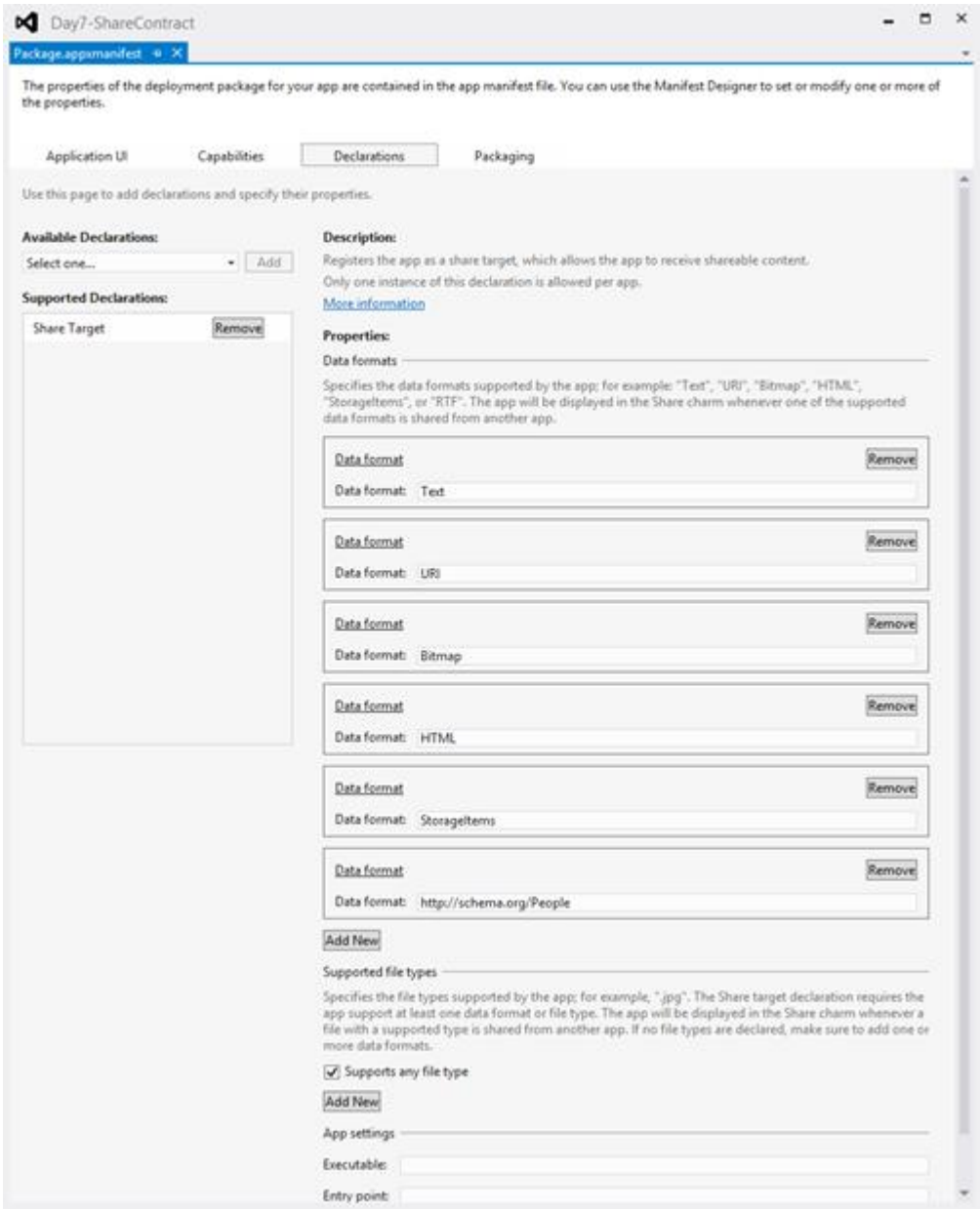


当你添加了 Share Target 之后，会有一些红色的 X 高亮显示。



因为单单添加 Share Target 声明是不够的。你还需要指定你的目标程序支持声

明类型的共享。在这里我填写了本文中涉及到的所有格式。下面这个显示了我设置完了 Share Target 声明的截图。



现在，我们进入 behind-code，在代码里面还有更多的事情需要处理。我的示例程序中将支持所有数据类型的共享和消费。所以涉及到的代码行数要多一点。

首先，打开 App.xaml.cs 文件，然后添加一个方法 OnShareTargetActivated，当共享目标被激活时，会被调用。

```
protectedoverridevoid OnShareTargetActivated(ShareTargetActivatedEventArgs args)
{
var rootFrame = newFrame();
      rootFrame.Navigate(typeof(MainPage), args.ShareOperation);
Window.Current.Content = rootFrame;
Window.Current.Activate();
}
```

上面的代码表示，但是共享目标程序被激活时，要加载哪个页面。这个文件不一定就是 MainPage，在这里我这是演示使用。

现在，下面是大段的逻辑代码。我为每种类型的判断都添加了注释，你可以关注你感兴趣的。下面的代码是在 MainPage.xaml.cs 文件的 OnNavigatedTo 函数中。你要记住的是，这样的逻辑处理一般都是在程序中单独写一个“SearchTarget”页面，来负责共享数据的获取。

```
ShareOperation share;
string title;
string description;
string text;
string customData;
string customDataFormat = "http://schema.org/People/";
string formattedText;
Uri uri;
IReadOnlyList<IStorageItem> storageItems;
IRandomAccessStreamReference bitmapImage;
IRandomAccessStreamReference thumbImage;


protectedoverrideasyncvoid OnNavigatedTo(NavigationEventArgs e)
{
      share = e.Parameter asShareOperation;

awaitTask.Factory.StartNew(async () =>
      {
            title = share.Data.Properties.Title;
            description = share.Data.Properties.Description;
```

```
            thumbImage = share.Data.Properties.Thumbnail;

//IF THERE WAS FORMATTED TEXT SHARED
if (share.Data.Contains(StandardDataFormats.Html))
            {
                formattedText = await share.Data.GetHtmlFormatAsync();
            }
//IF THERE WAS A URI SHARED
if (share.Data.Contains(StandardDataFormats.Uri))
            {
                uri = await share.Data.GetUriAsync();
            }
//IF THERE WAS UNFORMATTED TEXT SHARED
if (share.Data.Contains(StandardDataFormats.Text))
            {
                text = await share.Data.GetTextAsync();
            }
//IF THERE WERE FILES SHARED
if (share.Data.Contains(StandardDataFormats.StorageItems))
            {
                storageItems = await share.Data.GetStorageItemsAsync();
            }
//IF THERE WAS CUSTOM DATA SHARED
if (share.Data.Contains(customDataFormat))
            {
                customData = await share.Data.GetTextAsync(customDataFormat);
            }
//IF THERE WERE IMAGES SHARED.
if (share.Data.Contains(StandardDataFormats.Bitmap))
            {
                bitmapImage = await share.Data.GetBitmapAsync();
            }

//MOVING BACK TO THE UI THREAD, THIS IS WHERE WE POPULATE OUR INTERFACE.
await Dispatcher.RunAsync(CoreDispatcherPriority.Normal, async () =>
                {
                    TitleBox.Text = title;
                    DescriptionBox.Text = description;

if (text != null)
                        UnformattedTextBox.Text = text;
if (uri != null)
                    {
                        UriButton.Content = uri.ToString();
                        UriButton.NavigateUri = uri;
                    }

if (formattedText != null)
                        HTMLTextBox.NavigateToString(HtmlFormatHelper.GetStaticFragment(formattedText));

if (bitmapImage != null)
```

```
                {
IRandomAccessStreamWithContentType bitmapStream = awaitthis.bitmapImage.OpenReadAsync();
BitmapImage bi = newBitmapImage();
                    bi.SetSource(bitmapStream);
                    WholeImage.Source = bi;

                    bitmapStream = awaitthis.thumbImage.OpenReadAsync();
                    bi = newBitmapImage();
                    bi.SetSource(bitmapStream);
                    ThumbImage.Source = bi;
                }

if (customData != null)
                {
StringBuilder receivedStrings = newStringBuilder();
JsonObject customObject = JsonObject.Parse(customData);
if (customObject.ContainsKey("type"))
                    {
if (customObject["type"].GetString() == "http://schema.org/Person")
                        {
                            receivedStrings.AppendLine("Type: " + customObject["type"].Stringify());
JsonObject properties = customObject["properties"].GetObject();
                            receivedStrings.AppendLine("Image: " + properties["image"].Stringify());
                            receivedStrings.AppendLine("Name: " + properties["name"].Stringify());
                            receivedStrings.AppendLine("Affiliation: " + properties["affiliation"].Stringify());
                            receivedStrings.AppendLine("Birth Date: " + properties["birthDate"].Stringify());
                            receivedStrings.AppendLine("Job Title: " + properties["jobTitle"].Stringify());
                            receivedStrings.AppendLine("Nationality: " +
properties["Nationality"].Stringify());
                            receivedStrings.AppendLine("Gender: " + properties["gender"].Stringify());
                        }
                        CustomDataBox.Text = receivedStrings.ToString();
                    }
                }
            });
        });
}
```

如上面代码所示,每种类型的代码都有所不同，这取决于数据的类型。

1.4. 总结

今天，我们学习了如何将共享合约添加到程序中。共享合约提供了一种非常

酷的新模式，以在程序之间进行交互。希望你有机会在程序中使用共享合约。

你可以点击下面的图片下载到本文的示例代码:

明天,我将介绍存储数据，涉及到两方面：本地存储----存储在用户设备上，以及将数据漫游到云端。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