Windows 8 开发 31 日

第 22 日

使用 Play To

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 22 日使用 Play To

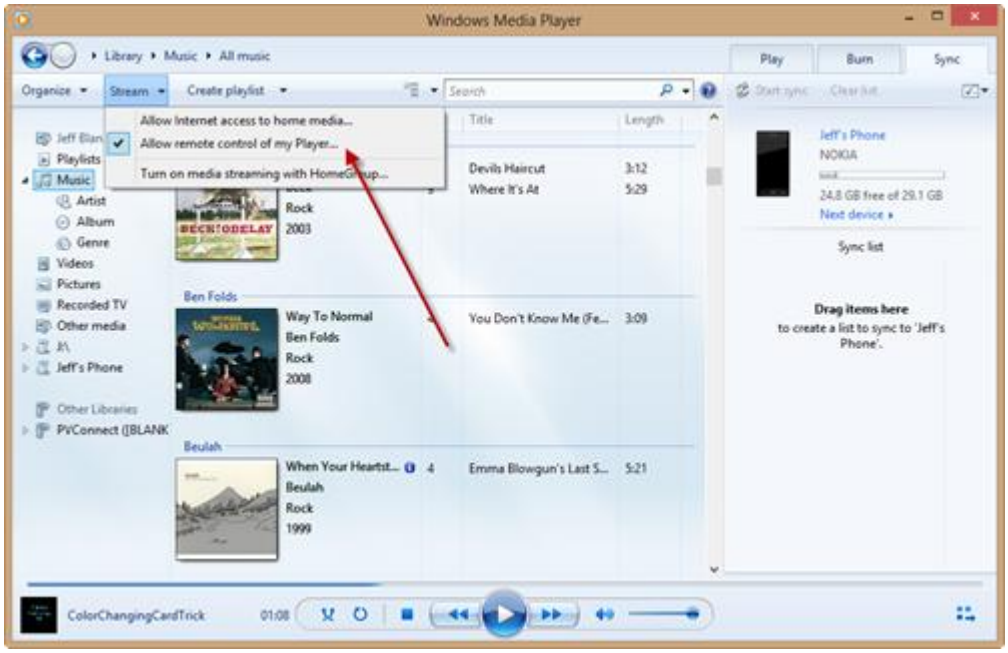
1.0. 介绍

今天，我来介绍一下 Windows 8 中另外一个非常酷的功能：Play To。它的核心特征是可以将内容（图片、音乐或视频）从你的电脑中分享到一台电视机、或者别的一台电脑，Xbox 360 等设备上。想象一下，当你在 YouTube 上发现了一个很好的视频，你想要将这个视频分享给屋子里面的别人。Play To 可以帮助你将视频分享到电视机上，那样就不用每个人都来围着你的电脑看。

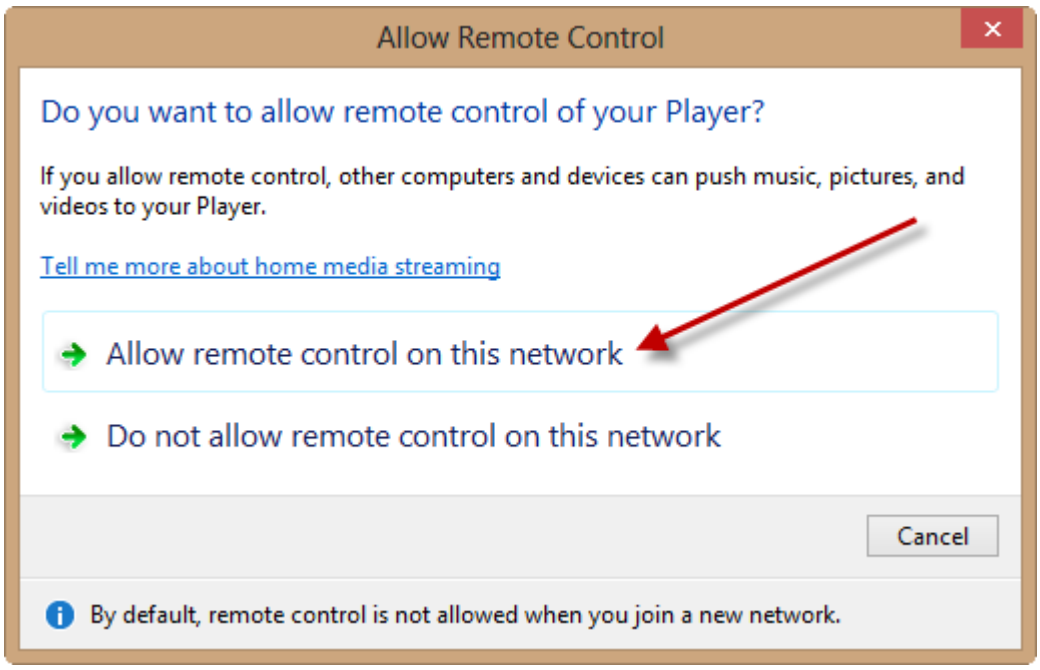
当我第一开始调查这个技术时，我考虑到它测试起来会有点困难。虽然在我房间里面有两台好看的 TVs，但是它们太老了，不能支持 Play To。（如果你想检查你的设备，请查阅 Microsoft's Compatibility Center.）。不管怎么样，我发现你只需要有另外一台可运行 Windows 8 的机器，或者一台 Xbox 360 即可。

我的配置是这样的，我在 Samsung Series 7 Slate（运行 Windows 8 Pro）上运行我的程序，可以在我的家庭网络中使用 Play To 将我硬盘中的内容分享到另外一台机器中的 Windows Media Player 上。同样我还测试了一下我的 Xbox 360，非常不错。

在开始编码之前，我需要确保你有一个方法来测试这个功能。在 Windows 8 中，代开 Windows Media Player 程序。如下图：



选择“Allow remote control of my Player…”选项。会有如下提示你进行选择：



选择“Allow remote control on this network”之后，你的机器就已经在你的网络中注册为一个可以使用 Play To 内容的设备了。

1.1. 在程序中添加 Play To 代码

相比于后面部分的代码，这部分代码是比较容易的。当我第一次学习如何实现的时候，我以为需要去 package.appxmanifest 文件中做一些修改，就像第 07 日中的共享一样。不过这次，我错了，不需要去修改这个文件。

首先，我们需要创建一个 PlayToManager 对象，借助该对象，可以通过 Play To 分享我们的内容。我还创建了一个 event handler，用来处理当用户连接到目标设备时的响应，最后，设置一下我们想要发送的内容。下面是完整的代码：

```
PlayToManager manager = null;
CoreDispatcher dispatcher = null;
```

```
protectedasyncoverridevoidOnNavigatedTo(NavigationEventArgs e)
{
dispatcher = Window.Current.CoreWindow.Dispatcher;
manager = PlayToManager.GetForCurrentView();
manager.SourceRequested += manager_SourceRequested;
}
```

```
voidmanager_SourceRequested(PlayToManager sender, PlayToSourceRequestedEventArgsargs)
{
var deferral = args.SourceRequest.GetDeferral();
var handler = dispatcher.RunAsync(CoreDispatcherPriority.Normal, () =>
      {
args.SourceRequest.SetSource(MusicSource.PlayToSource);
deferral.Complete();
      });
}
```

```
protectedasyncoverridevoidOnNavigatingFrom(NavigatingCancelEventArgs e)
{
manager.SourceRequested -= manager_SourceRequested;
}
```

上面就是所需代码。还记得第 20 日谈论的打印吗？超过了 400 行代码才打印出来。而在这里的代码并不是太多。

上面代码中，最神奇的地方应该就是设置 source 了。我获取 page 中 MediaElement 的 source。标准的 Play To 用例是用户在他的个人平板或者设备中看到内容，然后想将内容分享到别的设备上。

1.2. 把你的程序变为 Play To 目标

在本示例中，我将继续添加一个功能到程序中：我们的程序可以收到 Play To 内容，这些内容可能来自我们的家庭网络。为了完成这个功能，首先需要从 PlayToReceiver 开始，以及与之相应必须实现的许多 event handler。我修改了 OnNavigatedTo 方法：

```
PlayToManager manager = null;
CoreDispatcher dispatcher = null;
PlayToReceiver receiver = null;
```

```
protectedasyncoverridevoidOnNavigatedTo(NavigationEventArgs e)
{
dispatcher = Window.Current.CoreWindow.Dispatcher;
manager = PlayToManager.GetForCurrentView();
manager.SourceRequested += manager_SourceRequested;

receiver = newPlayToReceiver();
receiver.PlayRequested += receiver_PlayRequested;
receiver.PauseRequested += receiver_PauseRequested;
receiver.StopRequested += receiver_StopRequested;
```

```
receiver.MuteChangeRequested += receiver_MuteChangeRequested;
receiver.VolumeChangeRequested += receiver_VolumeChangeRequested;
receiver.TimeUpdateRequested += receiver_TimeUpdateRequested;
receiver.CurrentTimeChangeRequested += receiver_CurrentTimeChangeRequested;
receiver.SourceChangeRequested += receiver_SourceChangeRequested;
receiver.PlaybackRateChangeRequested += receiver_PlaybackRateChangeRequested;
receiver.SupportsAudio = true;
receiver.SupportsVideo = true;
receiver.SupportsImage = true;

receiver.FriendlyName = "Day #22 - PlayTo";

awaitreceiver.StartAsync();
}
```

如上所示，这里有 9 个 event handler 需要实现。我在这里特别强调，那是因为如果 9 个中，你没有全部都实现，那么你将不能调用 PlayToReceiver 的 StartAsync()方法。

至于 9 个方法的具体实现内容取决于你，下面是 9 个函数的介绍：

1.2.1. SourceChangeRequested

在这个方法中（可能是 9 个方法中最重要的一个），我们检测发送过来的 content 类型，获取出相应的 content，然后将其赋值给 UI 中适合的 XAML 控件。ShowSelectedPanel()方法用来管理 MediaElement 的显示与否以及 stop/start。

```
asyncvoidreceiver_SourceChangeRequested(PlayToReceiver sender, SourceChangeRequestedEventArgsargs)
{
if (args.Stream != null)
      {
if (args.Stream.ContentType.Contains("image"))
            {
awaitDispatcher.RunAsync(CoreDispatcherPriority.High, () =>
                  {
BitmapImage bmp = newBitmapImage();
bmp.SetSource(args.Stream);
PhotoSource.Source = bmp;
ShowSelectedPanel(1);
```

```
                });
            }
elseif (args.Stream.ContentType.Contains("video"))
            {
awaitDispatcher.RunAsync(CoreDispatcherPriority.High, () =>
                {
VideoSource.SetSource(args.Stream, args.Stream.ContentType);
ShowSelectedPanel(3);
                });
            }
elseif (args.Stream.ContentType.Contains("audio"))
            {
awaitDispatcher.RunAsync(CoreDispatcherPriority.High, () =>
                {
MusicSource.SetSource(args.Stream, args.Stream.ContentType);
ShowSelectedPanel(2);
MusicSource.Play();
                });
            }
        }
}
```

1.2.2. PlayRequested

这个以及所有的后续 event handler 都是简单的调用一下适当的方法(这里是调用“Play”)，然后通知我们的 source，调用已经发生了。

```
asyncvoidreceiver_PlayRequested(PlayToReceiver sender, objectargs)
{
awaitDispatcher.RunAsync(CoreDispatcherPriority.High, () =>
        {
MusicSource.Play();
VideoSource.Play();
receiver.NotifyPlaying();
        });
}
```

1.2.3. PauseRequested

几乎与 PlayRequested 完全相同，止步这里调用的是 Pause。

Practically identical to the PlayRequested method, this one implements Pause.

```
asyncvoidreceiver_PauseRequested(PlayToReceiver sender, objectargs)
{
awaitDispatcher.RunAsync(CoreDispatcherPriority.High, () =>
      {
MusicSource.Pause();
VideoSource.Pause();
receiver.NotifyPaused();
      });
}
```

1.2.4. 剩下的 event handler

如上面看到的，最后 5 个 event handler 都非常的类似。

```
privateasyncvoidreceiver_PlaybackRateChangeRequested(PlayToReceiver sender,
PlaybackRateChangeRequestedEventArgsargs)
{
awaitDispatcher.RunAsync(CoreDispatcherPriority.High, () =>
      {
VideoSource.PlaybackRate = args.Rate;
      });
}
```

```
privateasyncvoidreceiver_CurrentTimeChangeRequested(PlayToReceiver sender,
CurrentTimeChangeRequestedEventArgsargs)
{
awaitDispatcher.RunAsync(CoreDispatcherPriority.High, () =>
      {
if (VideoSource.CanSeek)
            {
                  {
VideoSource.Position = args.Time;
receiver.NotifySeeking();
                  }
            }
      });
}
```

```
privateasyncvoidreceiver_TimeUpdateRequested(PlayToReceiver sender, objectargs)
```

```
{
awaitDispatcher.RunAsync(CoreDispatcherPriority.High, () =>
        {
receiver.NotifyTimeUpdate(VideoSource.Position);
        });
}
```

```
privateasyncvoidreceiver_VolumeChangeRequested(PlayToReceiver sender,
VolumeChangeRequestedEventArgsargs)
{
awaitDispatcher.RunAsync(CoreDispatcherPriority.High, () =>
        {
VideoSource.Volume = args.Volume;
        });
}
```

```
privateasyncvoidreceiver_MuteChangeRequested(PlayToReceiver sender, MuteChangeRequestedEventArgsargs)
{
awaitDispatcher.RunAsync(CoreDispatcherPriority.High, () =>
        {
VideoSource.IsMuted = args.Mute;
        });
}
```

这样，我们的程序就实现了通过 Play To 发送和接收内容了。

1.3. 总结

今天我们学习了 Play To 协议，允许用户将媒体文件发送到另外一个设备上，也可以接收从另外一个设备上发送过来的媒体文件。

点击下图，下载本文示例代码：

明天，我将介绍使用传感器：Compass（罗盘）。明天将学习到关于罗盘的许多内容。到时候见！

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