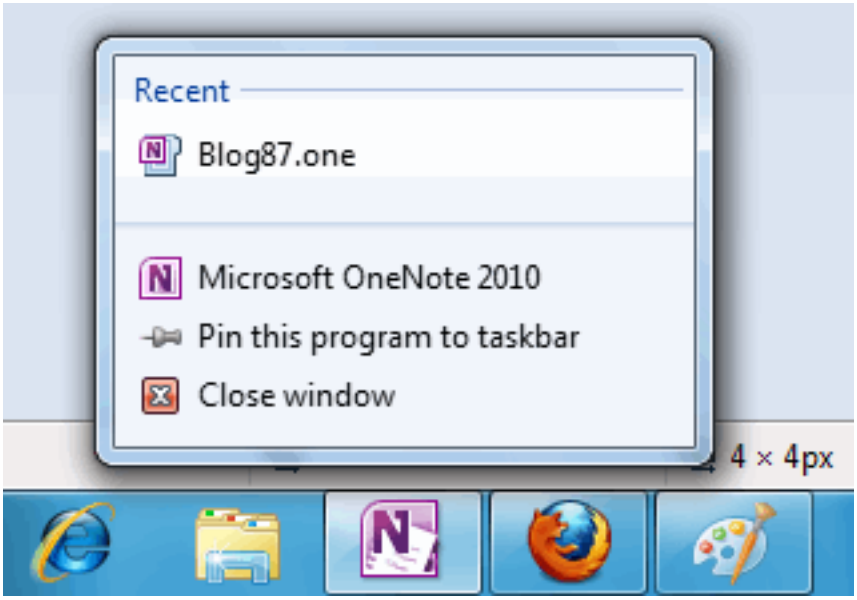
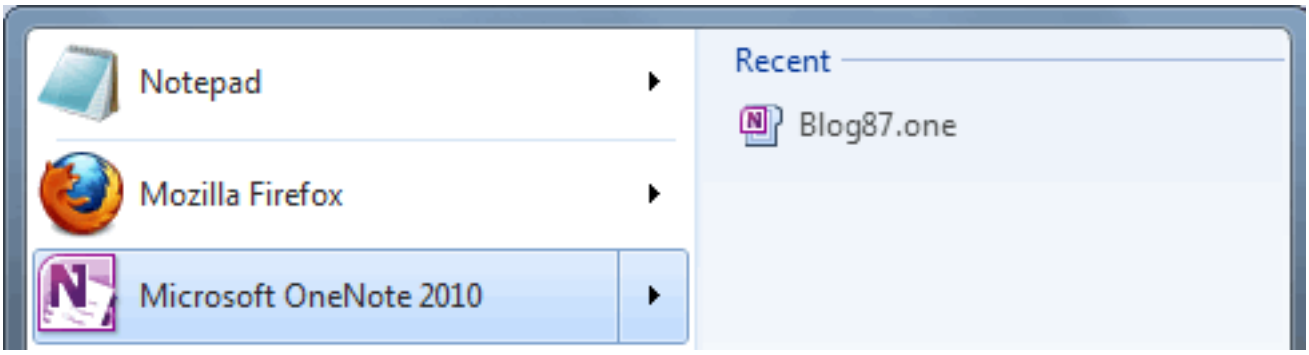
عنوان: **WPF4 و ویندوز 7 : به خاطر سپاری لیست آخرین فایل‌های گشوده شده توسط برنامه**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۰۹/۲۷ ۱۸:۲۳:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: WPF

اگر به برنامه‌های جدید نوشته شده برای ویندوز 7 دقت کنیم، از یک سری امکانات مخصوص آن جهت بهبود دسترسی پذیری به قابلیت‌هایی که ارائه می‌دهند، استفاده شده است. برای مثال برنامه‌ی OneNote مجموعه‌ی آفیس را در نظر بگیرید. اگر بر روی آیکون آن در نوار وظیفه‌ی ویندوز کلیک راست کنیم، لیست آخرین فایل‌های گشوده شده توسط آن مشخص است و با کلیک بر روی هر کدام، به سادگی می‌توان این فایل را گشود. یک چنین قابلیتی در منوی آغازین ویندوز نیز تعبیه شده است (شکل‌های زیر):





خبر خوب اینکه برای اضافه کردن این قابلیت به برنامه‌های WPF4 نیازی به کد نویسی نیست و این موارد که تحت عنوان استفاده از Jump list ویندوز 7 تعریف شده‌اند، با کمی دستکاری فایل App.Xaml برنامه، فعال می‌گردند:

```
<Application x:Class="WpfApplication1.App"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             StartupUri="MainWindow.xaml">
```

```
   <Application.Resources>
   </Application.Resources>
   <JumpList.JumpList>
       <JumpList ShowRecentCategory="True" />
   </JumpList.JumpList>
</Application>
```

همین! از این پس هر فایلی که توسط برنامه‌ی شما با استفاده از common file dialog boxes باز شود به صورت خودکار به لیست مذکور اضافه می‌گردد (بدیهی است Jump lists جزو ویژگی‌های ویندوز 7 است و در سایر سیستم عامل‌ها ندید گرفته خواهد شد).

سؤال: من اینکار را انجام دادم ولی کار نمی‌کنه!؟

پاسخ: بله. کار نمی‌کنه! این قابلیت تنها زمانی فعال خواهد شد که علاوه بر نکته‌ی فوق، پسوند فایل یا فایل‌هایی نیز به برنامه‌ی شما منتسب شده باشد. این انتساب‌ها مطلب جدیدی نیست و در تمام برنامه‌های ویندوزی باید توسط بکارگیری API ویندوز مدیریت شود. قطعه کد زیر اینکار را انجام خواهد داد:

```
using System;
using System.Runtime.InteropServices;
using Microsoft.Win32;

namespace Common.Files
{
  //from : http://www.devx.com/vb2themax/Tip/19554?type=kbArticle&trk=MSCP
  public class FileAssociation
  {
      const int ShcneAssocchanged = 0x8000000;
      const int ShcnfIdlist = 0;

      public static void CreateFileAssociation(
          string extension,
          string className,
          string description,
           string exeProgram)
      {
          // ensure that there is a leading dot
          if (extension.Substring(0, 1) != ".")
              extension = string.Format(".{0}", extension);

          try
          {
              if (IsAssociated(extension)) return;

              // create a value for this key that contains the classname
              using (var key1 = Registry.ClassesRoot.CreateSubKey(extension))
              {
                  if (key1 != null)
                  {
                      key1.SetValue("", className);
                      // create a new key for the Class name
                      using (var key2 = Registry.ClassesRoot.CreateSubKey(className))
                      {
                          if (key2 != null)
                          {
                              key2.SetValue("", description);
                              // associate the program to open the files with this extension
                              using (var key3 =
Registry.ClassesRoot.CreateSubKey(string.Format(@"{0}\Shell\Open\Command", className)))
                              {
                                  if (key3 != null) key3.SetValue("", string.Format(@"{0} ""%1""",
exeProgram));
                              }
                          }
                      }
                  }
              }

              // notify Windows that file associations have changed
              SHChangeNotify(ShcneAssocchanged, ShcnfIdlist, 0, 0);
          }
          catch (Exception ex)
          {
```

```
            //todo: log ...
        }
    }

    // Return true if extension already associated in registry
    public static bool IsAssociated(string extension)
    {
        return (Registry.ClassesRoot.OpenSubKey(extension, false) != null);
    }

    [DllImport("shell32.dll")]
    public static extern void SHChangeNotify(int wEventId, int uFlags, int dwItem1, int dwItem2);
  }
}
```

و مثالی از نحوه‌ی استفاده از آن:

```
private static void createFileAssociation()
{
   var appPath = Assembly.GetExecutingAssembly().Location;
    FileAssociation.CreateFileAssociation(".xyz", "xyz", "xyz File",
               appPath
               );
}
```

لازم به ذکر است که این کد در ویندوز 7 فقط با دسترسی مدیریتی قابل اجرا است (کلیک راست و اجرا به عنوان ادمین) و در سایر حالات با خطای Access is denied متوقف خواهد شد. به همین جهت بهتر است برنامه‌ی نصاب مورد استفاده این نوع انتسابات را مدیریت کند؛ زیرا اکثر آن‌ها با دسترسی مدیریتی است که مجوز نصب را به کاربر جاری خواهند داد. اگر از فناوری Click once استفاده می‌کنید به این مقاله و اگر برای مثال از NSIS کمک می‌گیرید به این مطلب مراجعه نمائید.

سؤال: من این کارها را هم انجام دادم. الان به چه صورت از آن استفاده کنم؟

زمانیکه کاربری بر روی یکی از این فایل‌های ذکر شده در لیست آخرین فایل‌های گشوده شده توسط برنامه کلیک کند، آدرس این فایل به صورت یک آرگومان به برنامه ارسال خواهد شد. برای مدیریت آن در WPF باید به فایل App.Xaml.cs مراجعه کرده و چند سطر زیر را به آن افزود:

```
public partial class App
  {
      public App()
      {
          this.Startup += appStartup;
      }

      void appStartup(object sender, StartupEventArgs e)
      {
          if (e.Args.Any())
          {
              this.Properties["StartupFileName"] = e.Args[0];
          }
      }
  //...
```

در این کد، e.Args حاوی مسیر فایل انتخابی است. برای مثال در اینجا مقدار آن به خاصیت StartupFileName انتساب داده شده است. این خاصیت در برنامه‌های WPF به صورت یک خاصیت عمومی تعریف شده است و در سراسر برنامه (مثلا در رخداد آغاز فرم اصلی آن یا هر جای دیگری) به صورت زیر قابل دسترسی است:

```
var startupFileName = Application.Current.Properties["StartupFileName"];
```

سؤال: برنامه‌ی من از OpenFileDialog برای گشودن فایل‌ها استفاده نمی‌کند. آیا راه دیگری برای افزودن مسیرهای باز شده به Jump lists ویندوز 7 وجود دارد؟

پاسخ: بله. همانطور که می‌دانید عناصر XAML با اشیاء دات نت تناظر یک به یک دارند. به این معنا که JumpList تعریف شده در ابتدای این مطلب در فایل App.XAML ، دقیقا معادل کلاسی به همین نام در دات نت فریم ورک است (تعریف شده در فضای نام System.Windows.Shell) و با کد نویسی نیز قابل دسترسی و مدیریت است. برای مثال:

```
var jumpList = JumpList.GetJumpList(App.Current);
var jumpPath = new JumpPath();
jumpPath.Path = "some path goes here....";
// If the CustomCategory property is null
// or Empty, the item is added to the Tasks category
jumpPath.CustomCategory = "Files";
JumpList.AddToRecentCategory(jumpPath);
jumpList.Apply();
```

به همین ترتیب، JumpPath ذکر شده در کدهای فوق، در کدهای XAML نیز قابل تعریف است:

```
<Application x:Class="Win7Wpf4.App"
           xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
           xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
           StartupUri="MainWindow.xaml">
  <Application.Resources>
  </Application.Resources>
  <JumpList.JumpList>
      <JumpList ShowRecentCategory="True">
          <JumpPath
               CustomCategory="Files"
               Path="Some path goes here..."
              />
      </JumpList>
  </JumpList.JumpList>
</Application>
```