XAML Do-It-Yourself シリーズ

第 13 回 ジャンプリスト

Microsoft®

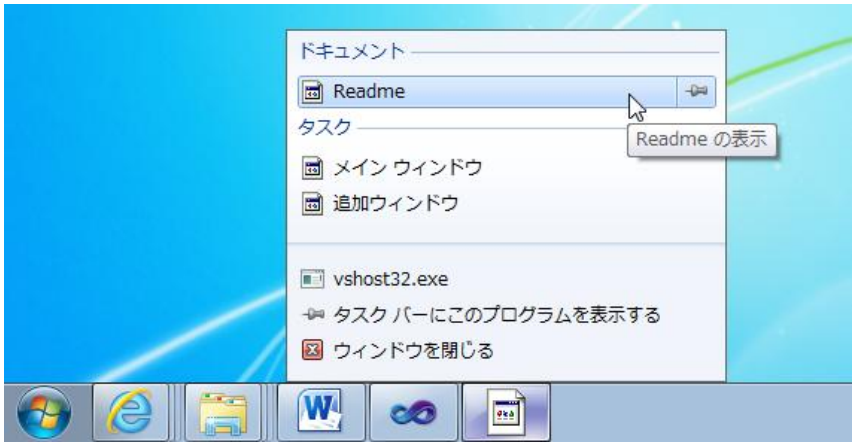
XAML Do-It-Yourself
第 13 回 ジャンプリスト

XAML Do-It-Yourself 第 13 回は、Windows 7 のジャンプリストにカスタム タスクを追加する方法について学習します。ジャンプリストは Windows 7 の新機能であり、タスクバー上のアイコンを右クリックすることで表示されます。WPF を使えば、ジャンプリストを簡単にカスタマイズできます。

今回は、次の 2 つを中心に学習します。

・ジャンプリストにカスタム タスクを追加する
・タスクから自分自身のアプリケーションを呼び出す

実際にジャンプリストを使用している様子を示します。ここで、ドキュメント カテゴリーと、タスク カテゴリーに項目が追加されています。



■XAML を使ったジャンプリストの追加

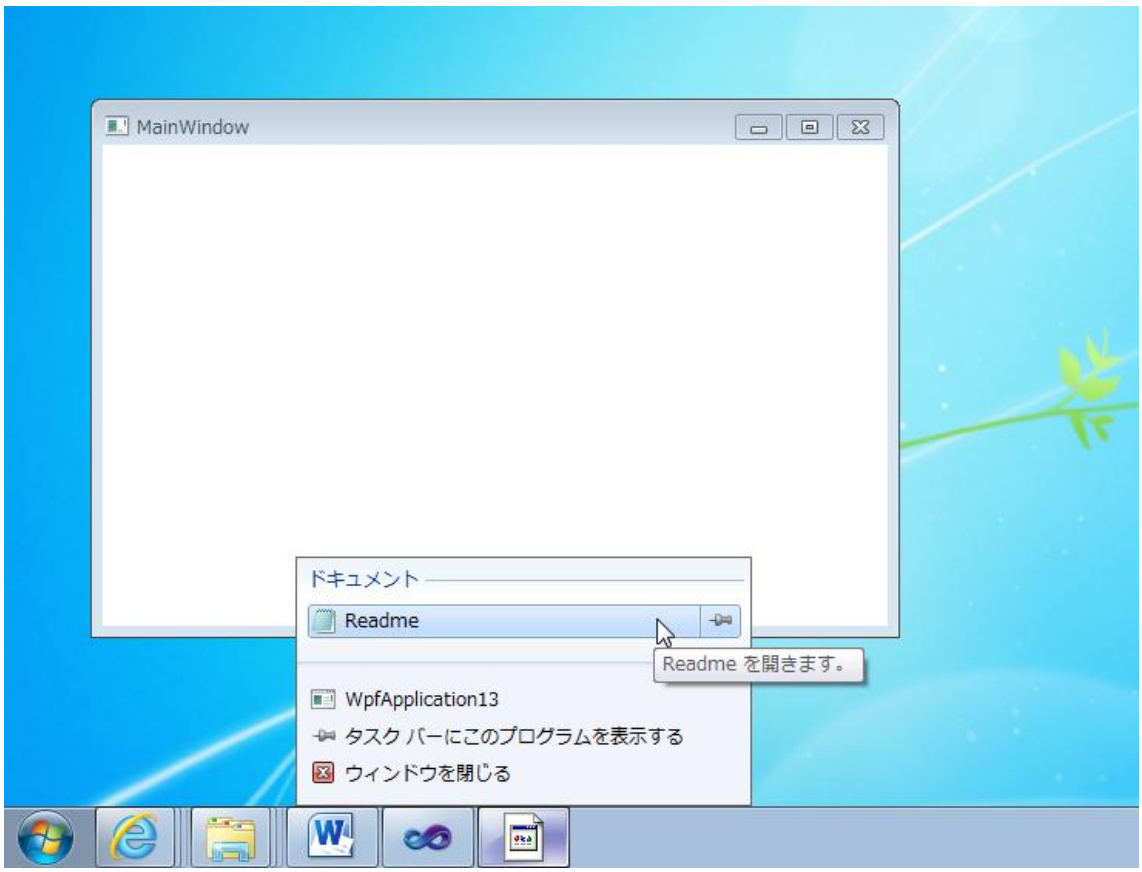
WPF では、ジャンプリストを扱うために System.Windows.Shell 名前空間にある JumpList クラスを使います。アプリケーション全体で共通するジャンプリストは、アプリケーション オブジェクトを定義する XAML（App.xaml）を使って、次のように記述できます。

```
<Application x:Class="WpfApplication13.App"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             StartupUri="MainWindow.xaml">
  <JumpList.JumpList>
    <JumpList>
      <JumpTask Title="Readme"
                Arguments="Readme.txt"
                Description="Readme を開きます。"
```

```
            CustomCategory="ドキュメント"
            ApplicationPath="notepad.exe"
            IconResourcePath="notepad.exe"
            WorkingDirectory="C:¥Users¥Public¥Documents" />
    </JumpList>
  </JumpList.JumpList>
</Application>
```

あらかじめユーザー¥パブリック¥パブリックのドキュメント フォルダーには Readme.txt を用意しておいてください。また、環境によっては、パスの設定を変更する必要があります。

プログラムを実行して、タスクバーでアイコンを右クリックすると次のように表示されます。



■プログラムによるジャンプリストの追加

ジャンプリストを XAML で定義する方法では、あらかじめ決まったフォルダーやファイル名しか設定できません。自分自身を呼び出す場合も含め、実際に呼び出したいアプリケーションや作業フォルダーを確実に設定するには、プログラムでフォルダーやファイル名を割り当てる必要があります。

上記の App.xaml と同じように処理するため、Application クラスを継承した App クラスで OnStartup イベントハンドラーを次のように定義できます。

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Windows;
using System.Windows.Shell;
using System.Reflection;
using System.IO;

namespace WpfApplication13
{
  /// <summary>
  /// App.xaml の相互作用ロジック
  /// </summary>
  public partial class App : Application
  {
    protected override void OnStartup(StartupEventArgs e)
    {
      JumpTask task = new JumpTask
      {
        Title = "Readme",
        Arguments = "Readme.txt",
        Description = "Readme を開きます。",
        CustomCategory = "ドキュメント",
        IconResourcePath =
Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.System),
"notepad.exe"),
        ApplicationPath =
Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.System),
"notepad.exe"),
        WorkingDirectory =
Environment.GetFolderPath(Environment.SpecialFolder.CommonDocuments),
      };

      JumpList list = new JumpList();
      list.JumpItems.Add(task);
      JumpList.SetJumpList(App.Current, list);
    }
  }
}
```

●自分自身のアプリケーションを呼び出す

ジャンプリストに登録するタスクは、自分自身のアプリケーションが処理したいことも多いでしょう。その際、どのような動作をさせるかをアプリケーションへの引数で指定することもできます。ここでは、メイン ウィンドウに引数で指定されたファイルを表示する機能を割り当ててみます。

まず、App クラスの OnStartup イベント ハンドラーを次のように記述します。

```
public partial class App : Application
{
  public static string DocToOpen = null;

  protected override void OnStartup(StartupEventArgs e)
  {
    if (e.Args.Count() > 0)
    {
      if (e.Args[0] == "/message")
      {
        MessageBox.Show("メッセージの表示");
        Shutdown();
      }

      // 引数をオープンするためのファイルとして保持
      DocToOpen = e.Args[0];
    }

    JumpTask task = new JumpTask
    {
      Title = "Readme",
      Arguments = "Readme.txt",
      Description = "Readme を開きます。",
      CustomCategory = "ドキュメント",
      IconResourcePath =
Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.System),
"notepad.exe"),
      ApplicationPath =
Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.System),
"notepad.exe"),
      WorkingDirectory =
Environment.GetFolderPath(Environment.SpecialFolder.CommonDocuments),
    };

    JumpTask task2 = new JumpTask
    {
      Title = "メッセージ表示",
      Arguments = "/message",
      Description = "メッセージを表示します",
      CustomCategory = "タスク",
      IconResourcePath = Assembly.GetEntryAssembly().CodeBase,
      ApplicationPath = Assembly.GetEntryAssembly().CodeBase,
      WorkingDirectory = Environment.CurrentDirectory,
    };

    JumpTask task3 = new JumpTask
    {
      Title = "Test document",
      Arguments = "TextFile.txt",
      Description = "テスト用のテキストを開きます",
      CustomCategory = "タスク",
      IconResourcePath = Assembly.GetEntryAssembly().CodeBase,
      ApplicationPath = Assembly.GetEntryAssembly().CodeBase,
      WorkingDirectory = Environment.CurrentDirectory,
```

```
    };

    JumpList list = new JumpList();
    list.JumpItems.Add(task);
    list.JumpItems.Add(task2);
    list.JumpItems.Add(task3);
    JumpList.SetJumpList(App.Current, list);
  }
}
```

まず、OnStartup イベント ハンドラーの第 2 引数に渡される e を使い、アプリケーションの引数（e.Args）を受け取ります。引数がある場合は、その引数が「/message」であれば、メッセージを表示するだけで終了し、そうでなければメイン ウィンドウでオープンするファイル名としてプロパティに設定します。

ジャンプリストには、「/message」という引数を渡すものと、「TextFile.txt」という引数を渡すものの 2 つのタスクを追加します。追加したタスクは、JumpTask オブジェクトの CustomCategory は指定した分類名ごとにまとめられてジャンプリストに表示されます。

引数としてオープンするファイル名が指定された場合、それをメイン ウィンドウに読み込むため、MainWindow.xaml は次のように記述します。

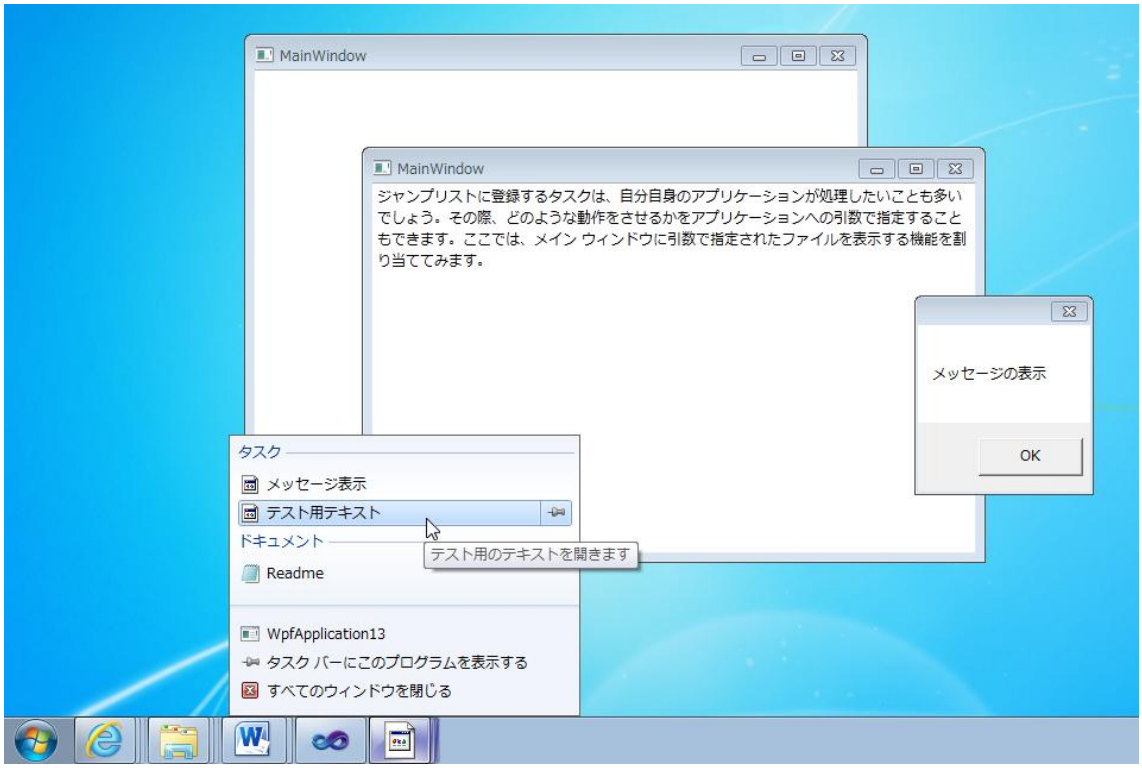
```
<Window x:Class="WpfApplication13.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="350" Width="525"
        Loaded="Window_Loaded">
  <Grid>
    <TextBox Name="textBox1" TextWrapping="Wrap" />
  </Grid>
</Window>
```

さらに、コード ハインドとして次のようなイベント ハンドラーを記述します。

```
private void Window_Loaded(object sender, RoutedEventArgs e)
{
  StreamReader reader = new StreamReader(App.DocToOpen);
  textBox1.Text = reader.ReadToEnd();
  reader.Close();
}
```

TextFile.txt を読み込ませるためには、アプリケーションの作成先にこのファイルを置いておく必要があります。ソリューション エクスプローラーでプロジェクト名を右クリックし、[追加] – [新しい項目] を選んで、TextFile.txt という「テキストファイル」を追加してください。追加した TextFile.txt のプロパティは、ビルド アクションを「なし」に、出力ディレクトリにコピーを「新しい場合はコピーする」に設定します。

プログラムを実行した様子を示します。



■まとめ

ジャンプリストは、Windows 7 の特徴的な機能のひとつであり、WPF を使うことで容易に実装できます。ジャンプリストはアプリケーション クラスだけでなく、ウィンドウ クラスにおいても追加することができます。すでにジャンプリストを構築している場合は、そのオブジェクトを使うか、JumpList.GetJumpList を使って設定されているジャンプリストを取り出して使ってください。