文章编号：2096-1472(2021)-09-44-04
DOI:10.19644/j.cnki.issn2096-1472.2021.09.010

多媒体管理工具的设计与实现

邹 珺

(苏州农业职业技术学院，江苏 苏州 215008)
✉zouj@szai.edu.cn

摘 要：当前很多软件系统的UI界面越来越多地使用了音视频、图片等多媒体文件，为了让用户方便、有效地管理多媒体文件，多媒体管理工具能对音视频、图片文件进行分类管理，能够查询文件，浏览、删除文件列表视图，查看文件详细信息，实现自定义的定制视图等功能。本文主要描述基于MVVM模式，使用WPF框架对多媒体管理工具进行开发和设计，通过面包屑视图控件实现视图导航，结合ListBox显示音视频、图片信息，实现多媒体管理工具的主要功能。结果表明，该工具强大的视觉设计特性实现了具有现代感的交互体验，满足了用户追求现代时尚的心理需求，也更好地实现了多媒体文件的有效管理。

关键词：多媒体管理工具；MVVM；WPF；面包屑
中图分类号：TP312 **文献标识码**：A

Design and Implementation of Multimedia Management Tools

ZOU Jun

(*Suzhou Agricultural Vocational College*, *Suzhou* 215008, *China*)
✉zouj@szai.edu.cn

Abstract: At present, more and more multimedia files such as audio, video and pictures are used in User Interface (UI) of many software systems. In order to make it convenient and effective for users to manage multimedia files, multimedia management tools can classify audio, video and picture files, query files, browse and delete file list views, view file details, and implement user-defined views. This paper mainly proposes to develop and design MVVM-based (Model-View-ViewModel) multimedia management tools using WPF (Windows Presentation Foundation) framework. Main functions of multimedia management tool are achieved by using breadcrumb view control to realize view navigation and ListBox to display audio, video and picture information. The results show that the powerful visual design features of the tool realize a modern interactive experience, meet the psychological needs of users in pursuit of modern fashion, and better realize effective management of multimedia files.

Keywords: multimedia management tools; MVVM; WPF; breadcrumbs

1 引言(Introduction)

WPF(Windows Presentation Foundation)的出现带来了桌面级应用软件的技术革新，传统的Windows Forms技术已经有些力不从心。微软渐渐地放弃了Windows Forms平台上的进一步开发，将重心转向了WPF上面。WPF为用户界面、2D/3D图形文档和媒体等提供了统一的描述和操作方法，不再像Windows Forms那样基于GDI+，而是基于DirectX 9/10技术，使用WPF开发的用户界面不仅具有漂亮的外观，而且还可以为用户界面应用3D效果。现代软件的UI已经不是十年前所能比拟的了，Windows 10的软件界面已经达到了绚丽的标准。如果要在UI界面上添加3D、音频或视频等功能，会耗费技术人员很多时间和精力，而且还达不到预期的效果，而利用WPF中的资源、样式、模板、数据绑定等技术，能够实现具有超绚效果的音乐图片管理工具，通过其强大的视觉设计特性来实现具有现代感的用户界面[1]。

2 多媒体管理工具概述(Overview of multimedia management tools)

多媒体管理工具的用户界面如图1所示。整个面板使用

Grid控件分成两行。最顶部的行放置转场用的指示控件，这是一个Expander控件，可以允许用户折叠面板，展开后会显示系统内置的四个转场效果的选择框。中间放置了一个转场控件，底部用一个自定义的滚动条控件来放置按钮。

图1 WPF多媒体管理工具用户界面

Fig.1 User interface of WPF multimedia management tool

首次进入该工具时，将显示一个空白的用户界面，用户可以单击标题栏的按钮显示音乐列表视图和图片列表视图，比如单击图片列表视图按钮后，将会在下面的面板上显示指定文件夹中的图片列表。用户可以单击其中的某幅图片查看图片详细信息，如图2所示。

图2 多媒体管理首页

Fig.2 Homepage of multimedia management

音乐视图提供了音乐专辑列表显示，当用户选中某个专辑图片时，会显示该专辑的音乐描述信息、音乐家以及专辑的详细信息。如图3所示，当选中图片列表中的图片时，会显示图片路径、大小以及访问日期等信息。在使用系统时，会看到当切换视图时，会具有动态的转场效果，同时在音乐专辑的封面切换到音乐信息的描述时，转场特效非常动感。面包屑控件能实现动感的用户导航面板[1]。

图3 使用面包屑控件导航视图

Fig.3 Navigate the view using the breadcrumb control

3 面包屑视图控件的实现(Implementation of breadcrumb view control)

3.1 面包屑管理器用户界面的实现

面包屑管理器用户控件，就是在用户主界面下面动感的视图导航控件，该控件类似Windows 10的面包屑导航效果。当开启了多种类型的视图控件时，会在导航面板上自动显示出视图类型，每种类型有一个新的按钮，并在按钮上显示出当前视图的个数[3]。当单击某个按钮，会显示视图预览列表框，单击某个小预览图标便会显示其对应的大预览图标，如图4所示。

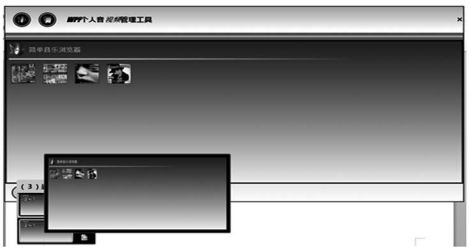

图4 面包屑管理服务面板

Fig.4 Breadcrumb management service panel

面包屑管理服务定义在一个单独的类库项目BreakCrumbControl中，在主用户界面上，面包屑管理器要作为一个容器，能够管理添加的视图控件。这些控件是实现了IbreadCrumbView接口的用户控件，面包屑自身又要能够根据这些加入的控件类型显示按钮让用户可以选择加入的控件列表。在BreadCrumbControl中，实现了一个用户控件BreadCrumbViewManager，这个控件将作为主窗体的显示控件，显示到主窗体Grid控件的ContentPresenter中。

BreadCrumbViewManager需要实现以下几个工作：

(1)能够被添加到其他的用户控件或窗体的视觉树中。

(2)用户能够创建自己的实现了IbreadCrumbView接口的控件，并显示在特定容器中。

(3)当一个新的视图控件被添加后，确保加载的视图控件被成功加入。

(4)确保有一个存在的控件类型可以使控件能被添加，如果存在一种类型的视图控件，一个视图控件将被包装为一个WrappedIBreadCrumbView对象并被添加到一个关联了指定类型的ObservableCollection<WrappedIBreadCrumbView>泛型集合中；如果不存在一种类型的视图控件，那么将新建一个新的类型加入字典中，并新建一个ObservableCollection<WrappedIBreadCrumbView>泛型集合来包含这个WrappedIBreadCrumbView视图[4]。

(5)管理器中的视图在进行切换时要具有动画转场特效。

3.2 使用转场控件实现转场效果

通过使用Transitionals.dll来实现转场特效，首先需要添加对于该程序集的引用。为了使XAML可以使用定义在其中的控件，需要在控件声明区添加对于该程序集的引用；然后在用户控件的资源定义区使用合并资源字典引入定义在Resources/AppStyles.xaml中的资源；最后定义一个具有三列的Grid[5]。XAML的定义代码如下：

```
<!—因为要用到转场特效，因此需要添加对于转场相关控件的程序集和命名空间的引用——>
<UserControl x:Class="BreadCrumbControl.BreadCrumbViewManager"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:local="clr-namespace:BreadCrumbControl"
xmlns:transitionals="clr-namespace:Transitionals;assembly=Transitionals"
xmlns:transitionalsControls="clr-namespace:
```

```
Transitionals.Controls;assembly=Transitionals"
    HorizontalAlignment="Stretch" VerticalAlignment="Stretch">
    <!--定义控件级别的资源-->
    <UserControl.Resources>
    <ResourceDictionary>
    <!--使用资源字典合并资源-->
    <ResourceDictionary.MergedDictionaries>
    <!--指定资源路径-->
    <ResourceDictionary Source="../Resources/AppStyles.xaml"/>
    </ResourceDictionary.MergedDictionaries>
    </ResourceDictionary>
    </UserControl.Resources>
    <!--定义三列布局-->
    <Grid>
    <Grid.RowDefinitions>
    <RowDefinition Height="Auto"/>
    <!--中间列将用来放置用户控件，需要具有最大显示比例-->
    <RowDefinition Height="*"/>
    <RowDefinition Height="Auto"/>
    </Grid.RowDefinitions>
```

3.3 使用滚动列表控件FrictionScrollViewer

在Grid的第三列，使用了FrictionScrollViewer自定义控件，在滚动条的内部，需要定义一个区域来放置按钮。按钮的数量是不固定的，可以通过将FrictionScrollViewer控件的ItemsSource属性绑定到控件视图列表来实现[6]。XAML的定义代码如下：

```
    <!—按钮显示区域的滚动条-->
    <local:FrictionScrollViewer x:Name="ScrollViewer" Grid.Row="2"
    Style="{StaticResource ScrollViewerStyle}">
    <!--将ItemsSource绑定到泛型列表-->
    <ItemsControl x:Name="items" ItemsSource="{Binding}">
    <!--指定容器面板-->
    <ItemsControl.ItemsPanel>
    <ItemsPanelTemplate>
    <!--定义虚拟化面板显示按钮-->
    <VirtualizingStackPanel IsItemsHost="True" Orientation="Horizontal" />
    </ItemsPanelTemplate>
    </ItemsControl.ItemsPanel>
    <!--定义列表项的模板-->
    <ItemsControl.ItemTemplate>
    <DataTemplate>
    <!--指定按钮控件的数据模板-->
    <ToggleButton x:Name="btn" Style="{StaticResource crumbButtonStyle}"
    Margin="15,5,15,5" ToolTip="{Binding Value[0].BreadCrumbItem.DisplayName}">
    <Grid>
```

3.4 定义面包屑管理器用户控件

因为在XAML中使用了大量的绑定，因此在类的构造函数中首要的工作是设置控件的DataContext属性来指定绑定集合，代码如下：

```
    public partial class BreadCrumbViewManager : UserControl
    {  //默认的转场类型
    private TransitionType currentTransitionType=TransitionType.FadeAndGrow;
    //转场效果映射集合
       private Dictionary<TransitionType, Transition>
       transitionsMap=new Dictionary<TransitionType, Transition>();
       //控件视图集合，实现观察者模式
       private new ObservableDictionary
       <Type, ObservableCollection<WrappedIBreadCrumbView>>
       crumbs=new ObservableDictionary
       <Type, ObservableCollection<WrappedIBreadCrumbView>>();
       public BreadCrumbViewManager()
       {  //指定控件的DataContext属性
                this.DataContext=crumbs;
                InitializeComponent();
                SetupTransitions();//初始化转场
       }
         ……//其他代码省略
       }
```

3.5 添加面包屑

在ViewModel中，ShowViewInBreadCrumbControl()方法是最常用来显示视图控件的方法。该方法在内部调用了BreadCrumbViewManager的AddCrumb()方法[7]。该方法的代码如下：

```
    public void AddCrumb(IBreadCrumbView newCrumb)
    {
       if (newCrumb !=null)//判断新视图的值
       {   //转换为视觉元素
           Visual visual=newCrumb as Visual;
           if (visual !=null)
           {   //指定转场的内容为新控件
            transitionBox.Content=newCrumb;
            //判断视图控件的值是否存在
            if (!crumbs.ContainsKey(newCrumb.GetType()))
            {   //如果不存在，则实例化一个新的ObservableCollection
                ObservableCollection<WrappedIBreadCrumb
```

```
View> localCrumbs=
            New ObservableCollection<WrappedIBreadCrumbView>();
            //将视图控件添加到ObservableCollection集合中
            localCrumbs.Add(CreateWrapper(newCrumb));
            //将该集合加到观察字典中
            crumbs.Add(newCrumb.GetType(), localCrumbs);
          }
          else
            {  //如果存在则直接加到观察字典中
            crumbs[newCrumb.GetType()].Add(CreateWrapper(newCrumb));
            }
          }
      }
    }
```

3.6 移除面包屑

每个缩略图右侧都具有两个按钮，这两个按钮用来移除或查看当前选中的视图，移除视图的代码定义在RemoveCrumb_Click事件处理代码中。其实现代码如下：

```
private void RemoveCrumb_Click(object sender, RoutedEventArgs e)
    {
      try
      {  //得到当前的要移除的WrappedIBreadCrumbView实例
          WrappedIBreadCrumbView crumbToRemove=
            (WrappedIBreadCrumbView)((Button)sender).Tag;
          //得到当前IBreadCrumbView对象实例，位于面板上
          IBreadCrumbView currentCrumbView=
          (IBreadCrumbView)transitionBox.Content;
          //得到当前要被移除的IBreadCrumbView实例
          IBreadCrumbView crumbToRemoveView=
          (IBreadCrumbView)crumbToRemove.BreadCrumbItem;
          //如果要移除的视图有一些变更
          if (crumbToRemoveView.IsDirty)
          {  //提示是否要立即保存
             if (MessageBox.Show("要移除的视图已经变化,可能对设置发生了改变 " +"\r\n你真的想移除吗","移除确认", MessageBoxButton.YesNo,
    MessageBoxImage.Question)==MessageBoxResult.Yes)
             {  //确定是否要移除当前视图
                CheckForCurrentCrumbAndConfirmRemoval(crumbToRemove,currentCrumbView, crumbToRemoveView);
             }
          }
          else
          {  //直接移除
            CheckForCurrentCrumbAndConfirmRemoval(crumbToRemove,currentCrumbView, crumbToRemoveView);
          }
          }
          catch
          {
                //异常处理代码
          }
        }
```

与删除视图相伴的是查看视图，该方法的实现较简单。从按钮的Tag属性中得到WrappedIBreadCrumbView对象，然后将转场对象的内容设置为BreadCrumbItem即可。其他的转场效果由transitionBox这个控件来完成[8]。

4 结论(Conclusion)

本文使用WPF实现了具有动感效果的多媒体管理器，界面部分采用WPF技术，包括3D旋转效果、动感的转场特效，同时结合了数据模板、样式和资源，核心部分主要通过面包屑管理控件，将多种视图控件添加到容器中，达到美观的效果。最终，实现了对图片、音频、视频等多媒体文件的系统化管理，使用户有更佳的体验效果。

参考文献(References)

[1] 李斌.基于WPF的图片预览控件的设计与实现[J].福建电脑,2018,34(5):120-121.

[2] 侯天峰,张志伟,葛陆蔚.基于WPF的图片浏览器设计与实现[J].微型电脑应用,2017,33(4):53-55.

[3] 刘珊珊,赵庆,曹豹,等.基于WPF的油藏模型三维可视化解决方案[J].西安石油大学学报(自然科学版),2021,36(1):73-79.

[4] 尚旭明,张立成.基于WPF的三维仿真系统的研究与应用[J].计算机技术与发展,2016,26(9):39-42.

[5] 张继梅.如何使用蒙版的技巧美化多媒体影视作品[J].电脑知识与技术,2020,16(2):221-223.

[6] 陈广山.基于WPF的UI设计模式研究[J].鸡西大学学报,2016,16(8):32-35.

[7] UTAMA A Z, JANG D S. Development of UML tool using WPF framework and forced-directionality graph algorithm[J]. Journal of Korea Multimedia Society, 2019, 22(6):706-715.

[8] 霍晓钢.数字媒体系统开发中基于WPF的行为的应用[J].计算机时代,2020,38(8):61-64.

作者简介：

邹　珺(1981-)，女，硕士，讲师.研究领域：软件开发，系统测试.