现代操作系统应用开发实验报告

学号： 14970011　　**班级：** 2015 级教务 2 班

姓名： 宋思亭　　**实验名称：** homework3

一．参考资料

Windows UWP Namespaces：https://docs.microsoft.com/en-us/uwp/api/

[WPF]静态资源(StaticResource)和动态资源(DynamicResource)：

课件、Demo

二．实验步骤

1. 阅读所给 demo 以及本次实验要求，在第二周作业的基础上进行修改完善

2. 添加并实现 TodoItem 类（列出主要成员函数）

```
public event PropertyChangedEventHandler PropertyChanged;
private string _title;
private ImageSource _img;
public string details;
public DateTime date;
public bool? Finish;
public TodoItem(DateTime date, ImageSource img, string title = "", string details = "", bool finish = false);
private void NotifyPropertyChanged(string propertyname);
public string title;
public ImageSource img;
public void UpdateItem(DateTime date, ImageSource img, string title, string details);
```

3. 添加和实现类TodoItemViewModel（列出主要成员函数）

```
private ObservableCollection<Models.TodoItem> items;
private Models.TodoItem selectedItem;
public Models.TodoItem SelectedItem;
public TodoItemViewModel();
```

```
public ObservableCollection<Models.TodoItem> getItems;
public void AddItem(DateTime date, ImageSource img, string title, string detail);
```

4. 修改MainPage结构并添加VisualStat控制显示

控制宽度大于800时显示InlineToDoItemViewGrid

```
<VisualStateManager.VisualStateGroups>
    <VisualStateGroup x:Name="VisualStateGroup">
        <VisualState x:Name="VisualStateMin0">
            <VisualState.Setters>
                <Setter Target="InlineToDoItemViewGrid.(UIElement.Visibility)"
Value="Collapsed" />
                <Setter Target="delete.Visibility" Value="Collapsed" />
                <Setter Target="ToDoListView.(Grid.ColumnSpan)" Value="2" />
            </VisualState.Setters>
            <VisualState.StateTriggers>
                <AdaptiveTrigger MinWindowWidth="1" />
            </VisualState.StateTriggers>
        </VisualState>
        <VisualState x:Name="VisualStateMin800">
            <VisualState.StateTriggers>
                <AdaptiveTrigger MinWindowWidth="800" />
            </VisualState.StateTriggers>
        </VisualState>
    </VisualStateGroup>
</VisualStateManager.VisualStateGroups>
```

控制宽度小于600时隐藏图片

```
<VisualStateGroup>
    <VisualState x:Name="narrow">
        <VisualState.StateTriggers>
            <AdaptiveTrigger MinWindowWidth="0" />
        </VisualState.StateTriggers>
        <VisualState.Setters>
            <Setter Target="image.Visibility" Value="Collapsed" />
        </VisualState.Setters>
    </VisualState>
    <VisualState x:Name="VisualStateMin600">
        <VisualState.StateTriggers>
            <AdaptiveTrigger MinWindowWidth="600" />
        </VisualState.StateTriggers>
        <VisualState.Setters>
            <Setter Target="image.Visibility" Value="Visible" />
        </VisualState.Setters>
```

```
            </VisualState>
        </VisualStateGroup>
```

5. 将ListView改为模板，实现数据绑定

```
            <ListView x:Name="ToDoListView" Margin="20" IsItemClickEnabled="True"
ItemClick="itemClick" ItemsSource="{x:Bind ViewModel.getItems}">
                <ListView.ItemTemplate>
                    <DataTemplate x:DataType="md:TodoItem">
                        <UserControl>
                            <Grid>
                                <Grid.ColumnDefinitions>
                                    <ColumnDefinition Width="42"/>
                                    <ColumnDefinition Width="Auto"/>
                                    <ColumnDefinition Width="*"/>
                                    <ColumnDefinition Width="100"/>
                                </Grid.ColumnDefinitions>
                                <CheckBox x:Name="TodoCheckBox" Grid.Column="0"
VerticalAlignment="Center" Height="32" Width="32" Grid.ColumnSpan="2"
HorizontalAlignment="Right" Margin="0,34,24,34" Checked="checkBox"
Unchecked="uncheckBox" IsChecked="{x:Bind finish, Mode=TwoWay}"/>
                                <Image x:Name="image" Grid.Column="1" Source="{x:Bind img,
Mode=TwoWay}" Height="90" Width="90" Margin="0,3,12,7"/>
                                <TextBlock Grid.Column="2" Text="{x:Bind title,
Mode=TwoWay}" VerticalAlignment="Center" Foreground="Black" FontWeight="Normal"
FontSize="15" LineHeight="20" TextWrapping="Wrap" />
                                <Line Grid.Column="2" Stretch="Fill" Stroke="Black"
StrokeThickness="2" X1="1" VerticalAlignment="Center" HorizontalAlignment="Stretch"
Visibility="{Binding IsChecked, ElementName=TodoCheckBox, Converter={StaticResource
BoolToVisConverter}}"/>
                                <AppBarButton Grid.Column="3" Icon="Setting"
IsCompact="True" VerticalAlignment="Center">
                                    <AppBarButton.Flyout>
                                        <MenuFlyout>
                                            <MenuFlyoutItem Text="Edit"/>
                                            <MenuFlyoutItem Text="Delete"/>
                                        </MenuFlyout>
                                    </AppBarButton.Flyout>
                                </AppBarButton>
                            </Grid>
                        </UserControl>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
```

其中涉及到Line与CheckBox的绑定，增加类BoolToVisConverter

```
public object Convert(object value, Type targetType, object parameter, string
language);
public object ConvertBack(object value, Type targetType, object parameter, string
language);
```

6. 实现List的添加、删除、修改

```
private async void CreatButton_Click(object sender, RoutedEventArgs e)
{
    if (title.Text == "")
    {
        var i = new MessageDialog("Title can not be empty!").ShowAsync();
    }
    if (details.Text == "")
    {
        var i = new MessageDialog("Detail can not be empty!").ShowAsync();
    }
    if (date.Date.CompareTo(DateTime.Today) < 0)
    {
        var i = new MessageDialog("The due date has passed!").ShowAsync();
    }
    if (title.Text != "" && details.Text != "" &&
date.Date.CompareTo(DateTime.Today) >= 0)
    {
        if (createButton.Content.ToString() == "Create")
        {
            this.ViewModel.AddItem(date.Date.DateTime, pic.Source, title.Text,
details.Text);
            CancelButton_Click(null, null);
            ViewModel.SelectedItem = null;
            await new MessageDialog("Create successfully!").ShowAsync();
            Frame.Navigate(typeof(MainPage), "");
        }
        else
        {
            this.ViewModel.SelectedItem.UpdateItem(date.Date.DateTime,
pic.Source, title.Text, details.Text);
            await new MessageDialog("Update successfully!").ShowAsync();
            Frame.Navigate(typeof(MainPage), "");
        }
    }
}
private async void CancelButton_Click(object sender, RoutedEventArgs e)
{
    title.Text = "";
```

```
            details.Text = "";
            date.Date = System.DateTime.Now;
            RandomAccessStreamReference img =
RandomAccessStreamReference.CreateFromUri(new Uri("ms-appx:///Assets/fruit.jpg"));
            IRandomAccessStream stream = await img.OpenReadAsync();
            Windows.UI.Xaml.Media.Imaging.BitmapImage bmp = new
Windows.UI.Xaml.Media.Imaging.BitmapImage();
            bmp.SetSource(stream);
            pic.Source = bmp;
        }
```

7. 复用MainPage中部分结构代码，完成第NewPage的添加、删除、修改以及跳转

8. 调试项目

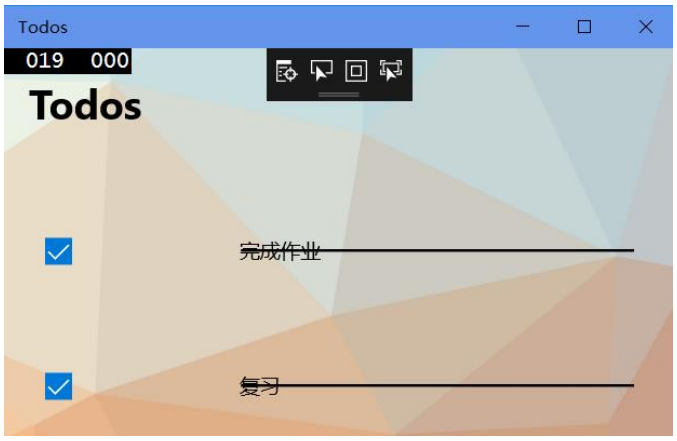
三．实验结果截图

1. 宽度与界面设计

大于 800 显示两列

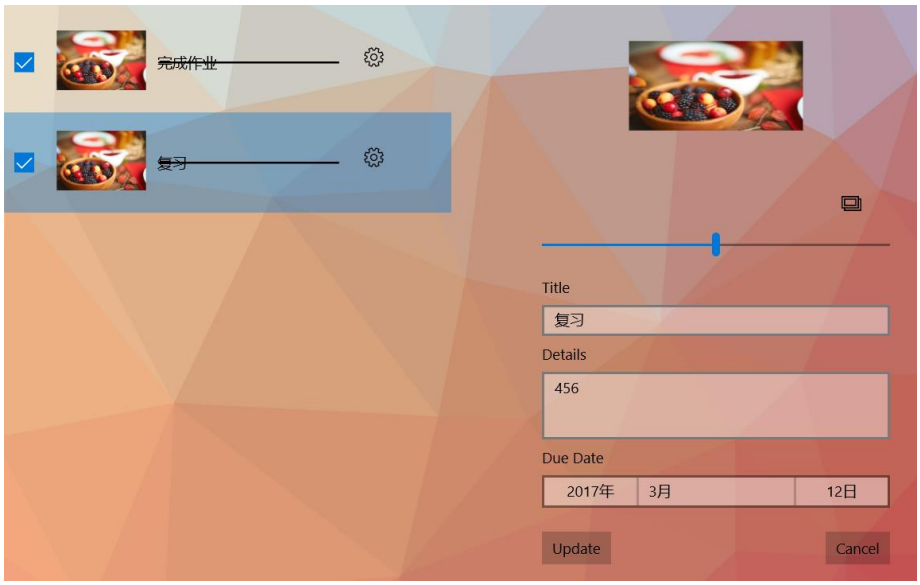
大于 600 显示一列

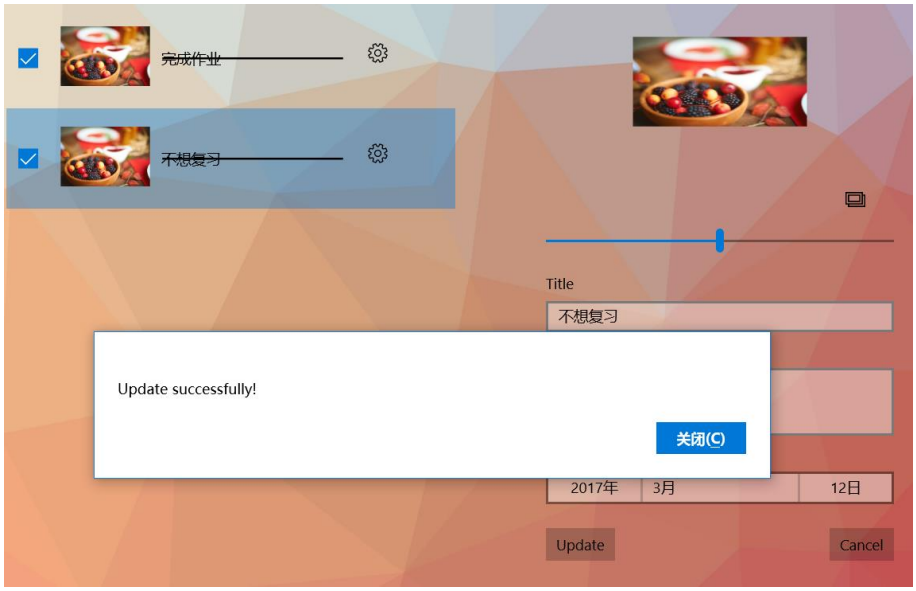
500 不显示图片



2. 添加、删除、修改

修改

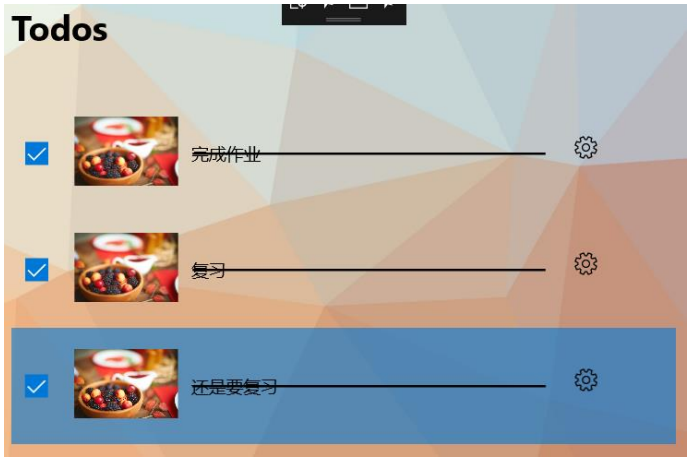



完成作业
不想复习
Title
不想复习
Update successfully!
关闭(C)
2017年
3月
12日
Update
Cancel

删除


Todos
完成作业
Title
Details
Due Date
2017年
3月
12日
Create
Cancel

添加



3. 点击左上角或底部返回按钮回到 MainPage

四．实验过程遇到的问题

1. 新建 Models 中的类后，外部调用无效，重新部署后解决
2. 窄页面修改 List 时，NewPage 页打开为空，修改.cs 文件中 OnNavigatedTo 函数后解决
3. 绑定 CheckBox 和 Line 查了很多 API 和博客
4. 数据绑定还是很迷

5. 稍复杂的页面逻辑还不完善

五．思考与总结

把实验先拆分成几个小部分，分步完成，效率更高而且能快速定位到出 bug 的地方。

VisualStateGroup 响应式布局使得一个页面具有多种状态。