

现代操作系统应用开发实验报告

学号： 15331046

班级： 软工一班

姓名： 陈志扬

实验名称： HW3-Todos

一．参考资料

请在这里列出对本实验有帮助你所参考的资料或者网站。

资料：2017week3 课件、2017-homework3v2.pptx 课件、2017-homework3-demo

2017-week3-demo

网站：<http://stackoverflow.com/> <http://blog.csdn.net/> <https://www.baidu.com/>

<https://www.cnblogs.com/>

<https://developer.microsoft.com/zh-cn/windows/apps/develop>

[https://docs.microsoft.com/zh-cn/windows/uwp/xaml-platform/xaml-syntax-guid](https://docs.microsoft.com/zh-cn/windows/uwp/xaml-platform/xaml-syntax-guide)

[e](#)

<https://docs.microsoft.com/zh-cn/windows/uwp/xaml-platform/>

<https://social.msdn.microsoft.com/forums/zh-cn/home> 等等

二．实验步骤

请在这里简要写下你的实验过程。

① 仔细阅读学习课件上的内容，查找有关 Adaptive UI 和 Data Binding 的资料，参考微软官方文档

② 进行 HW2 的移植，参考 week3-demo 和 homework3-demo，主要对后者各种接口和类逐步理解，熟悉其作用以及用处，再者，实现对窄屏和宽屏显示内容的要

求，具体用到 VisualStateManager、VisualState、Setter 等内容

- ③ 由于一开始对窗口宽度小于 600 不知如何处理，所以这一步先跳过，接下来根据宽屏下点击+号不作用，故在 MainPage.xaml.cs 文件中加入窗口宽度的判断。

若 Window.Current.Bounds.Width 小于 800，则点击+号可跳转到 NewPage。

代码如下：

```
private void AddAppBarButton_Click(object sender, RoutedEventArgs e)
{
    // 只能在窄屏下点击+号可以实现跳转
    if (Window.Current.Bounds.Width <= 800)
        Frame.Navigate(typeof(NewPage), ViewModel);
}
```

- ④ 接下来实现 AddTodoItem、RemoveTodoItem、UpdateTodoItem 方法，开始了一波搜索，AddTodoItem 和 RemoveTodoItem 可直接调用 Collection<T>.Add 和 Collection<T>.Remove 这两个 API，然而，对于 UpdateTodoItem，好像是用到 INotifyPropertyChanged，但由于查找了很久都不知道怎么实现（可能是感觉实现这个参考代码好像有点多，搞得有点害怕），所以我采用了先 Remove 在这个位置的 TodoItem，然后在这个位置 Add 更新后的 TodoItem。

代码如下：

```
public void UpdateTodoItem(string title, string description, System.DateTimeOffset set_day)
{
    // DIY
    // 获取TodoItem的下标
    var index = this.allItems.IndexOf(this.selectedItem);
    // 设置title, description, day
    this.selectedItem.title = title;
    this.selectedItem.description = description;
    this.selectedItem.day = set_day;
    // 删掉原来的TodoItem
    this.allItems.Remove(this.SelectedItem);
    // 在原位置插入更新update后的TodoItem
    this.allItems.Insert(index, this.selectedItem);
    // set selectedItem to null after update
    this.selectedItem = null;
}
```

- ⑤ 在 MainPage.xaml.cs 和 NewPage.xaml.cs 文件中分别实现 CreateButton_Clicked、CancelButton_Clicked、UpdateButton_Clicked 方法，Create、Update 成功后 Title、Details、Date 需要重置，并补充一点交互（创建、更新是否成功给出提示信息）
- ⑥ MainPage.xaml.cs 中实现对 list（点击一个 list，即点击一个 TodoItem）的操作 TodoItem_ItemClicked

```
private void TodoItem_ItemClicked(object sender, ItemClickEventArgs e)
{
    ViewModel.SelectedItem = (Models.TodoItem)(e.ClickedItem);
    // 当处于窄屏时, 跳转到NewPage
    if (Window.Current.Bounds.Width <= 800)
    {
        Frame.Navigate(typeof(NewPage), ViewModel);
    }
    else
    {
        Title.Text = ViewModel.SelectedItem.title;
        Details.Text = ViewModel.SelectedItem.description;
        DatePicker.Date = ViewModel.SelectedItem.day; // 加上日期
        CreateButton.Content = "Update"; // , 因为是点击TodoItem 所以对应u
    }
}
```

- ⑦ NewPage.xaml.cs 中实现 DeleteButton_Clicked 删除 TodoItem。

```
private void DeleteButton_Clicked(object sender, RoutedEventArgs e)
{
    if (ViewModel.SelectedItem != null)
    {
        ViewModel.RemoveTodoItem();
        Frame.Navigate(typeof(MainPage), ViewModel);
    }
}
```

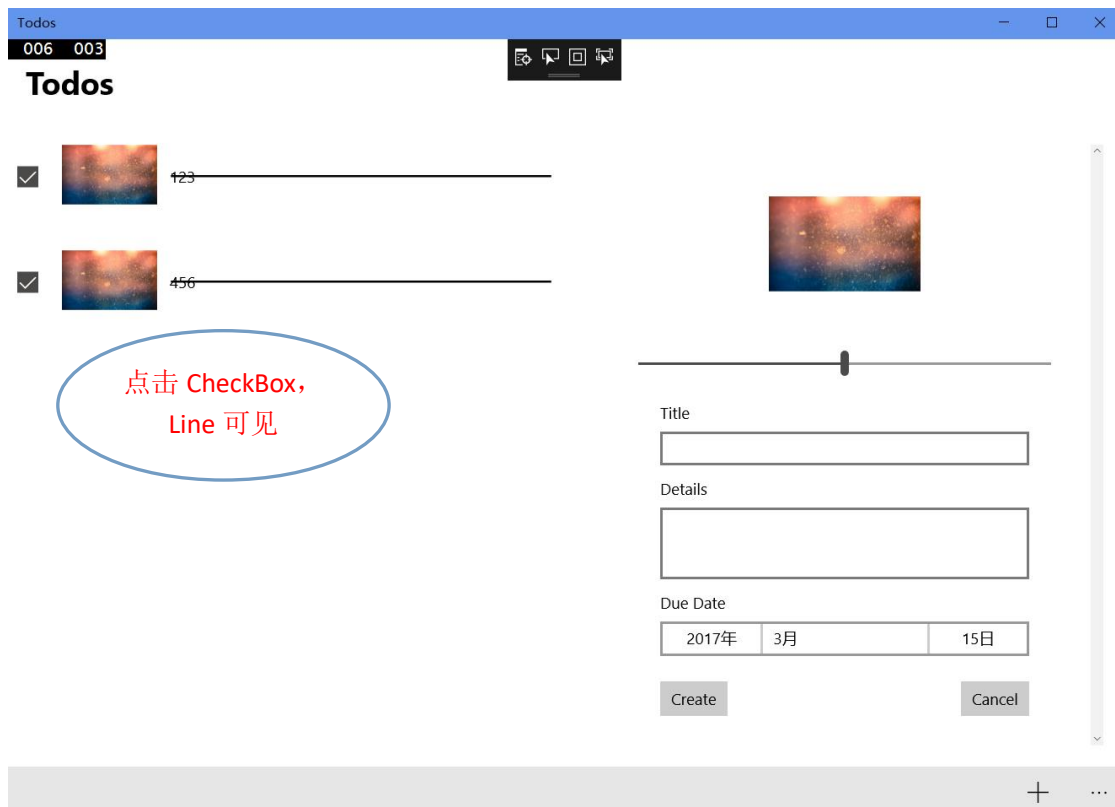
- ⑧ 主要功能都已实现, 需要添加对日期的处理, 这里我简单采用了这样的设计

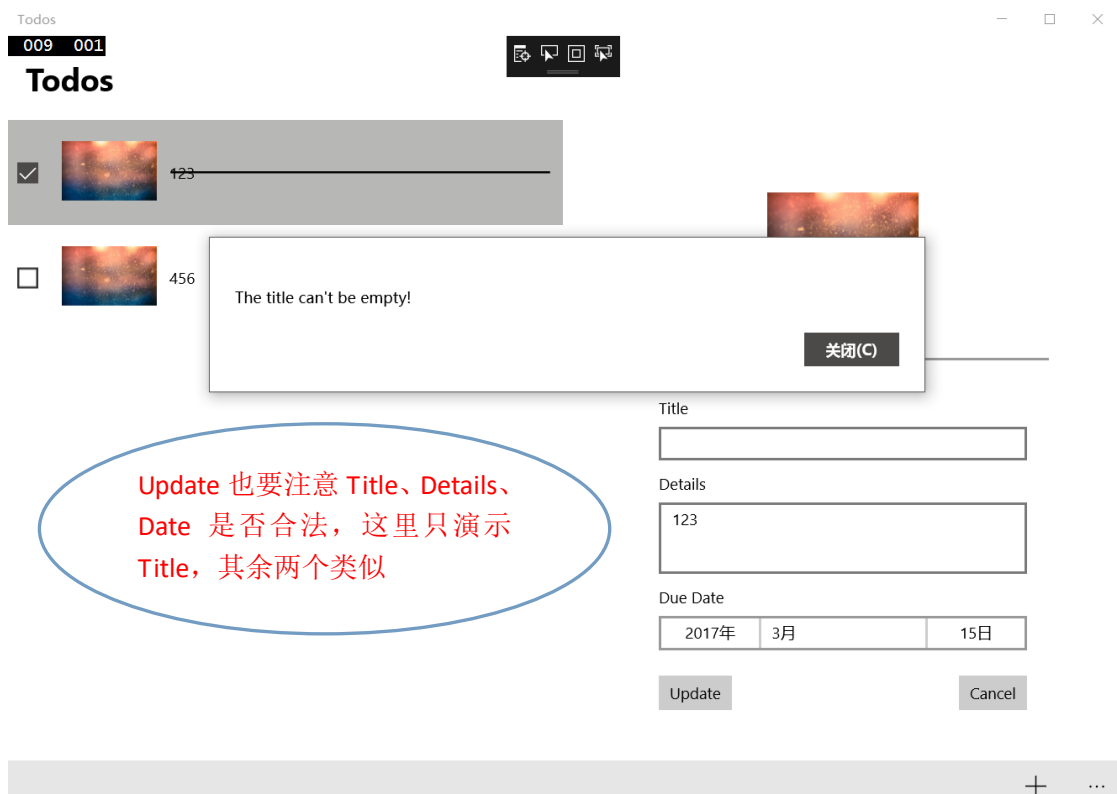
```
public System.DateTimeOffset day;
// 添加了形参 System.DateTimeOffset set_day, 用来表示用户设置的时间
public TodoItem(string title, string description, System.DateTimeOffset set_day)
{
    this.id = Guid.NewGuid().ToString();
    this.title = title;
    this.description = description;
    this.completed = false; // 默认为未完成
    this.day = set_day;
    this.date = day.ToString();
}
```

- ⑨ 至此, 再来弄图片显示, 根据 Stack Overflow 上的答案, 可知根据 UserControl 来处理 DataTemplate 的控件, 详细代码见报告第四部分【实验过程遇到的问题】。
- ⑩ 最后, 对 CheckBox 和 Line 的实现真的是“路漫漫其修远兮”, 但是我也真是“吾将上下而求索”, 对 Line 的 Visibility 绑定, 再根据 CheckBox 的 IsChecked 判断 Visible 或者 Collapsed。详细代码见报告第四部分【实验过程遇到的问题】。

三. 实验结果截图

请在这里把实验所得的运行结果截图。





Todos

011 001

Todos

☒  123

☐  456

There is no change! Please update one of them!

关闭(C)

Details

123

Due Date

2017年3月15日

UpdateCancel


如果信息没有改变，则提示无法更新

Todos

012 001

Todos

☐  I have been updated.

☐  456

Title

Details

Due Date

2017年3月15日


CreateCancel

123 已经被更新
I have been updated.
更新后右边 Title、Details、Date 重置

006000


Todos

☐




I have been updated.

☐




456

☐



I'm just created.

创建一个 Item，成功后右边 Title、Details、Date 重置



Title

Details

Due Date

2017年

3月

15日

Create

Cancel

009005

Todos

☐



I have been updated.

☐



456

☐



I'm just created.

点击这个 Item，右边信息显示出来，包括日期的改变



Title

I'm just created.

Details

I'm just created.

Due Date

2017年

4月

17日

Update

Cancel

←

Todos

—

□

×

001 000


+

↶

□

↷

Edit Todo Item



select

Title

I'm created in the newpage.

Details

I'm created in the newpage.

Due Date

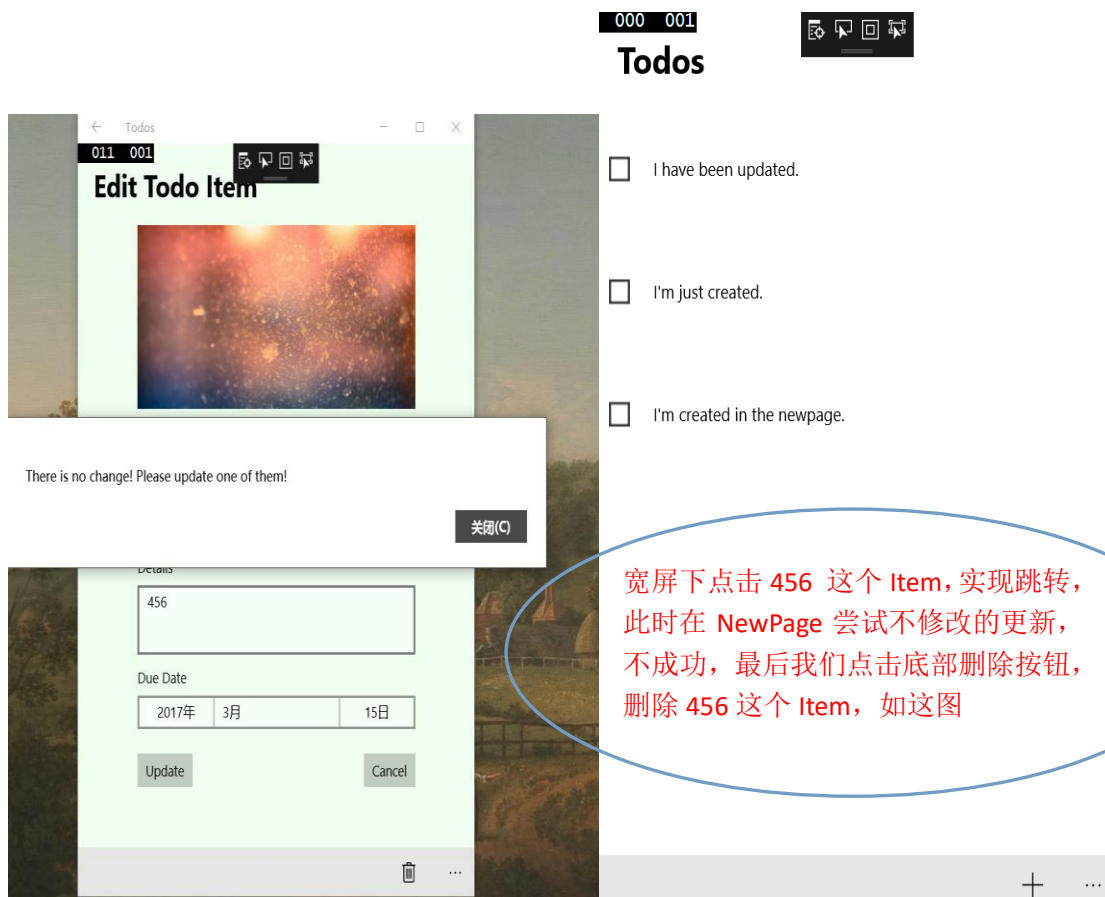
2018年3月15日

CreateCancel

🗑

...

窄屏下点击+号，实现
跳转，此时在 **NewPage**
创建一个 **Item**，注意
日期



011 001

Todos



☐ I have been updated.

☐ I'm just created.

☐ I'm created in the newpage.

☐ C#

☐ Happy Birthday!

窄屏状态下隐藏图片




000 000

Todos



☐  I have been updated.

☐  I'm just created.

☐  I'm created in the newpage.

☐  C#

☐  Happy Birthday!

☐  7



Title

C#

Details

How to learn C#

Due Date

2017年

3月

16日

Update

Cancel



四．实验过程遇到的问题

请在这里写下你在实验过程中遇到的问题以及解决方案。

- ① 加入了 DataTemplate 后窗口宽度小于 600 时图片不显示。在 Stack Overflow 上看到相关的问题，知道用 UserControl 来处理这个问题，但是一直不行，总在宽屏下隐藏了图片，而窄屏下却显示出图片，尝试了各种 Visible 和 Collapsed，最后，发现原来是宽度之间有个过渡问题，把 MinWindowWidth="1"Collapsed 的相关代码写进 UserControl 即可解决。

代码如下：

```
<DataTemplate x:DataType="md:TodoItem">
    <UserControl>
        <Grid Height="100">
            <!--设置小于600不显示图片-->
            <VisualStateManager.VisualStateGroups>
                <VisualStateGroup>
                    <VisualState x:Name="VisualStateMin0">
                        <VisualState.Setters>
                            <Setter Target="Image.(UIElement.Visibility)" Value="Collapsed"/>
                        </VisualState.Setters>
                        <VisualState.StateTriggers>
                            <AdaptiveTrigger MinWindowWidth="1"/>
                        </VisualState.StateTriggers>
                    </VisualState>
                    <VisualState x:Name="VisualStateMin600">
                        <VisualState.Setters>
                            <Setter Target="Image.(UIElement.Visibility)" Value="Visible"/>
                        </VisualState.Setters>
                        <VisualState.StateTriggers>
                            <AdaptiveTrigger MinWindowWidth="600"/>
                        </VisualState.StateTriggers>
                    </VisualState>
                </VisualStateGroup>
            </VisualStateManager.VisualStateGroups>
        </Grid>
    </UserControl>
</DataTemplate>
```

- ② 同样的 DataTemplate 问题，添加了 DataTemplate 后，控件的 Name 属性不能在 cs 文件里起作用，导致 CheckBox 和 Line 无法解决。数据绑定解决此问题，绑定 Line 的 Visibility 属性 加上 Path、ElementName 和 Converter 在 TodoItem.cs 文件里添加类 Converter 并实现相应的接口

IValueConverter.Convert(object value, Type targetType, object parameter, string language)

xaml 代码如下

```

<Grid.Resources>
    <md:Converter x:Key="Converter"/>
</Grid.Resources>
<CheckBox x:Name="CheckBox" Grid.Column="0" VerticalAlignment="Center" Height="32" Width="32" />
<Image x:Name="Image" Grid.Column="1" Source="Assets/background.jpg" Height="90" Width="90" Margin="0,3,12,7"/>
<TextBlock x:Name="Title" Text="{x:Bind title}" Grid.Column="2" VerticalAlignment="Center" Foreground="Black" FontWeight="Normal" FontSize="15" />
<Line x:Name="Line" Visibility="{Binding Path=IsChecked, ElementName=CheckBox, Converter={StaticResource Converter}}" Grid.Column="2" Stretch="

```

C#代码如下：

```

// 转换器，实现控制Line的Visibility
class Converter : IValueConverter
{
    public object Convert(object value, Type targetType, object parameter, string language)
    {
        bool myValue = (bool)value;
        if (myValue)
        {
            return Visibility.Visible;
        }
        else
        {
            return Visibility.Collapsed;
        }
    }
    public object ConvertBack(object value, Type targetType, object parameter, string language)
    {
        throw new NotImplementedException();
    }
}

```

③ 日期的问题。由于 demo 给的数据是 string date，对其处理将会比较困难，故我

自己添加了 DateTimeOffset，然后在对应的函数接口加上该形参，以便修改日期

数据。

```

// 日期
public System.DateTimeOffset day;
// 添加了形参 System.DateTimeOffset set_day，用来表示用户设置的时间
public TodoItem(string title, string description, System.DateTimeOffset set_day)
{

```

④ UpdateTodoItem 在宽屏下实时修改的问题。如果没用

INotifyPropertyChanged 接口，大多只是采用了 MainPage 导航到 MainPage

的方法，这样可以实现修改，但是这个“鸡肋且不友好”的跳转真是糟糕。所以，

我想了一个折中的方法，即记录下要 Update 的 Item 的位置，然后删除原来的

Item，再在这个位置插上更新后的 Item，这样既不影响美观实际，也能实现更新

需求。主要用到 Collection<T>.Indexof(T)、Collection<T>.Remove(T) 和

Collection<T>.Insert(int, T)

C#代码如下：

```
public void UpdateTodoItem(string title, string description, System.DateTimeOffset set_day)
{
    // DIY
    // 获取TodoItem的下标
    var index = this.allItems.IndexOf(this.selectedItem);
    // 设置title, description, day
    this.selectedItem.title = title;
    this.selectedItem.description = description;
    this.selectedItem.day = set_day;
    // 删掉原来的TodoItem
    this.allItems.Remove(this.SelectedItem);
    // 在原位置插入更新update后的TodoItem
    this.allItems.Insert(index, this.selectedItem);
    // set selectedItem to null after update
    this.selectedItem = null;
}
```

⑤ 在 MainPage.xaml.cs 和 NewPage.xaml.cs 中 ,CancelButton_Clicked 函数也要

注意一个问题，如果选中一个 Item，此时是要执行更新函数，但是如果你这时候点击了取消，虽然 Button 的 Content 会修改为“Create”，但其实执行的还是 Update 函数，所以得加上下面这个，取消后设置没有选中一个 Item，即为 null。

```
private void CancelButton_Clicked(object sender, RoutedEventArgs e)
{
    Title.Text = "";
    Details.Text = "";
    DatePicker.Date = DateTime.Now.Date;
    CreateButton.Content = "Create";
    ViewModel.SelectedItem = null; // 这非常重要，如果没有，会出现只是修改成Create，其实执行的函数还是update
}
```

⑥ 在 MainPage.xaml.cs 和 NewPage.xaml.cs 中，Create 和 Update 的判断问题，

究竟是执行 Create 还是 Update 函数。这个我用了一个判断，如果没有选中 Item 即是 Create，如果选中了那就执行 Update 函数。另外，在处理信息合法时，如果要 Update 的 Item 都不修改时，则不执行 Update 函数，同时告知用户需要至少更改一个信息才能 Update。

C#代码如下：

```

private void CreateButton_Clicked(object sender, RoutedEventArgs e)
{
    // check the textbox and datapicker
    // if ok
    // 判断是否选择了TodoItem
    // 没有选择的话进行Create
    if (ViewModel.SelectedItem == null)
    {
        // 同样信息不能为空或者有误
        if (Title.Text == "")
        {
            var i = new MessageDialog("The title can't be empty!").ShowAsync();
        }
        else if (Details.Text == "")
        {
            var i = new MessageDialog("The details can't be empty!").ShowAsync();
        }
        else if (DatePicker.Date < DateTime.Now.Date)
        {
            var i = new MessageDialog("The date before current date is wrong!").ShowAsync();
        }
        else
        {
            ViewModel.AddTodoItem(Title.Text, Details.Text, DatePicker.Date); // 调用TodoItemViewModel中的AddTodoItem方法
            Frame.Navigate(typeof(MainPage), ViewModel); // 跳转回MainPage
        }
    }
    // 选择了的话进入Update
    else
    {
        UpdateButton_Clicked(sender, e);
    }
}

private void UpdateButton_Clicked(object sender, RoutedEventArgs e)
{
    if (ViewModel.SelectedItem != null)
    {
        // check then update
        // 同样信息不能为空或者有误
        if (Title.Text == "")
        {
            var i = new MessageDialog("The title can't be empty!").ShowAsync();
        }
        else if (Details.Text == "")
        {
            var i = new MessageDialog("The details can't be empty!").ShowAsync();
        }
        else if (DatePicker.Date < DateTime.Now.Date)
        {
            var i = new MessageDialog("The date before current date is wrong!").ShowAsync();
        }
        // 若全部没修改, 则不能update
        else if (Title.Text == ViewModel.SelectedItem.title && Details.Text == ViewModel.SelectedItem.description && DatePicker.Date == ViewModel.SelectedItem.day)
        {
            var i = new MessageDialog("There is no change! Please update one of them!").ShowAsync();
        }
        else
        {
            ViewModel.UpdateTodoItem(Title.Text, Details.Text, DatePicker.Date); // 调用TodoItemViewModel中的UpdateTodoItem方法
            Frame.Navigate(typeof(MainPage), ViewModel);
        }
    }
}

```

⑦ TodoItemViewModel.cs 文件中 AddTodoItem、RemoveTodoItem 的实现。

```

public void AddTodoItem(string title, string description, System.DateTimeOffset set_day)
{
    this.allItems.Add(new Models.TODOItem(title, description, set_day)); // 直接调用Add方法
}

public void RemoveTodoItem()
{
    // DIY
    this.allItems.Remove(this.SelectedItem); // 直接调用Remove方法
    // set selectedItem to null after remove
    this.selectedItem = null;
}

```

⑧ NewPage.xaml.cs 文件中 DeleteButton_Clicked 的实现

```

private void DeleteButton_Clicked(object sender, RoutedEventArgs e)
{
    if (ViewModel.SelectedItem != null)
    {
        ViewModel.RemoveTodoItem();
        Frame.Navigate(typeof(MainPage), ViewModel);
    }
}

```

五 . 思考与总结

请在这里写下你本次试验的心得体会以及所思所想。

心得体会：通过本次作业，学习了 Adaptive UI 和 Data Binding，数据绑定是非常重要的，将会影响到接下来的课程学习，需要好好理解数据绑定。每一次的作业需要查找的文档很多，痛苦的是，文档很多很杂，看起来毫无头绪，但坚持下来一定会有所收获，坚持，继续努力！Fighting！

1. 实验报告提交格式为 pdf。
2. 实验内容不允许抄袭，我们要进行代码相似度对比。如发现抄袭，按 0 分处理。