现代操作系统应用开发实验报告

姓名：________刘宇庭___________

学号：_______16340158________

实验名称：____UWP 实验 1_____

一、参考资料

1. 博客
2. QQ 群提问答疑
3. 微软官方文档

二、实验步骤

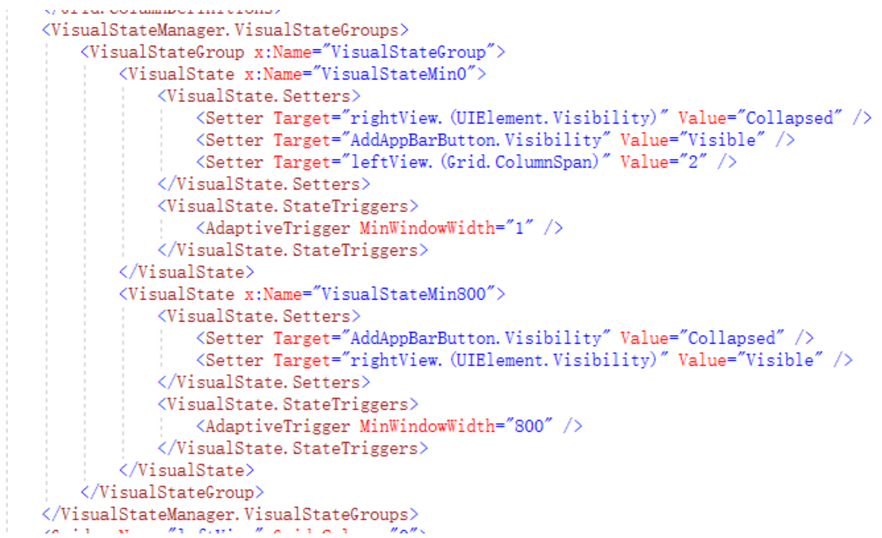
1. 自适应 UI 设计
2. Page 之间数据绑定
3. 逻辑结构部分完成优化
4. 页面设计美化

三、关键步骤截图

1. 自适应 UI

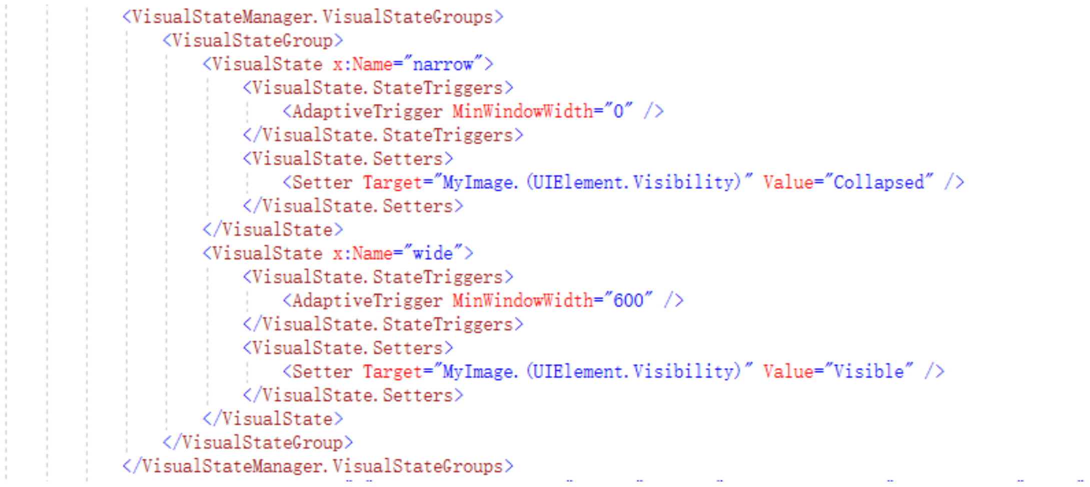
自适应分为两部分设计

① Mainpage 的宽度在 0 到 800 的时候右边部分不显示，大于 800 的时候右边部分显示



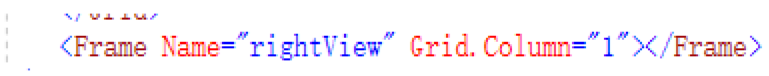
```
<VisualStateManager.VisualStateGroups>
    <VisualStateGroup x:Name="VisualStateGroup">
        <VisualState x:Name="VisualStateMin0">
            <VisualState.Setters>
                <Setter Target="rightView.(UIElement.Visibility)" Value="Collapsed" />
                <Setter Target="AddAppBarButton.Visibility"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

② Mainpage 的宽度在 0 到 600 的时候右 Item 的图片不显示，大于 600 的时候图片显示



```
<VisualStateManager.VisualStateGroups>
    <VisualStateGroup>
        <VisualState x:Name="narrow">
            <VisualState.StateTriggers>
                <AdaptiveTrigger MinWindowWidth="0" />
            </VisualState.StateTriggers>
            <VisualState.Setters>
                <Setter Target="MyImage.(UIElement.Visibility)" Value="Collapsed" />
            </VisualState.Setters>
        </VisualState>
        <VisualState x:Name="wide">
            <VisualState.StateTriggers>
                <AdaptiveTrigger MinWindowWidth="600" />
            </VisualState.StateTriggers>
            <VisualState.Setters>
                <Setter Target="MyImage.(UIElement.Visibility)" Value="Visible" />
            </VisualState.Setters>
        </VisualState>
    </VisualStateGroup>
</VisualStateManager.VisualStateGroups>
```

③ Mainpage 添加一个 Frame，让其宽度超过 800 显示

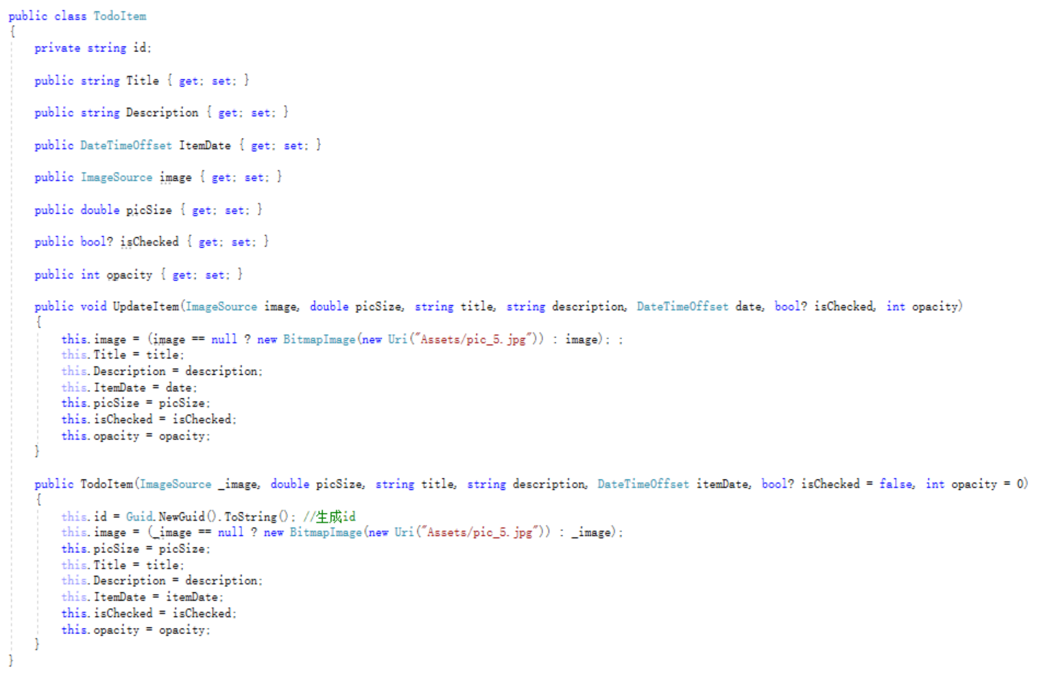


```
<Frame Name="rightView" Grid.Column="1"></Frame>
```

2. 数据绑定

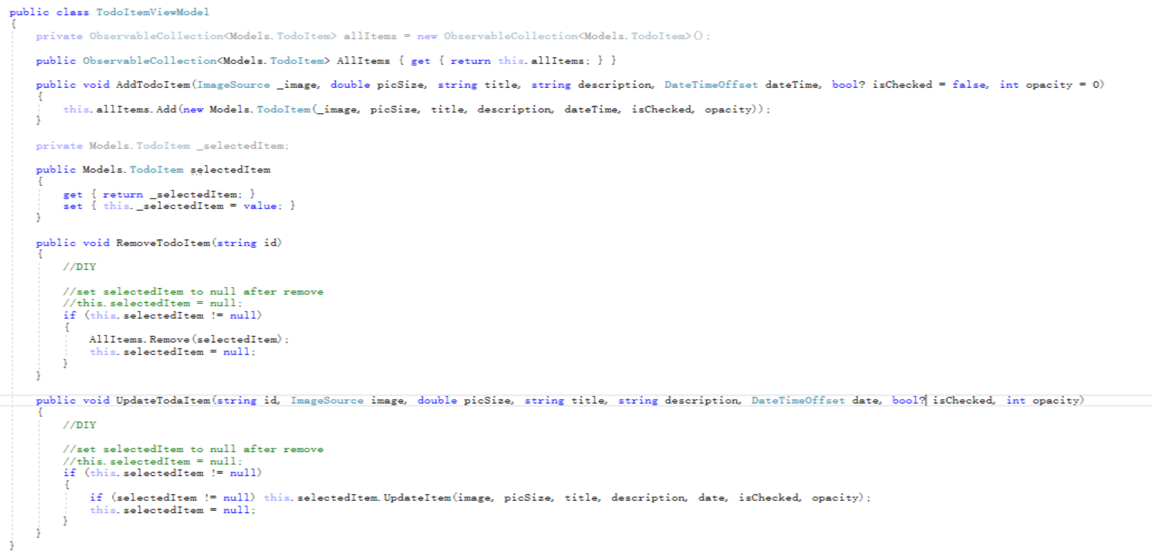
定义 Model

其中 isChecked 表示 checkbox 是否选定和 opacity 表示直线是否显示



```
public class TodoItem
{
    private string id;

    public string Title { get; set; }

    public string Description { get; set; }

    public DateTimeOffset ItemDate { get; set; }

    public ImageSource image { get; set; }

    public double picSize { get; set; }

    public bool? isChecked { get; set; }

    public int opacity { get; set; }

    public void UpdateItem(ImageSource image, double picSize, string title, string description, DateTimeOffset date, bool? isChecked, int opacity)
    {
        this.image = (image == null ? new BitmapImage(new Uri("Assets/pic_5.jpg")) : image); ;
        this.Title = title;
        this.Description = description;
        this.ItemDate = date;
        this.picSize = picSize;
        this.isChecked = isChecked;
        this.opacity = opacity;
    }

    public TodoItem(ImageSource _image, double picSize, string title, string description, DateTimeOffset itemDate, bool? isChecked = false, int opacity = 0)
    {
        this.id = Guid.NewGuid().ToString(); //生成id
        this.image = (_image == null ? new BitmapImage(new Uri("Assets/pic_5.jpg")) : _image);
        this.picSize = picSize;
        this.Title = title;
        this.Description = description;
        this.ItemDate = itemDate;
        this.isChecked = isChecked;
        this.opacity = opacity;
    }
}
```

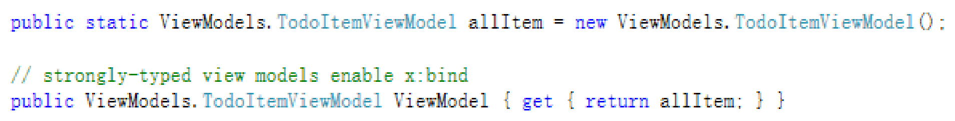
ModelList 中定义一些方法



```
public class TodoItemViewModel
{
    private ObservableCollection<Models.TodoItem> allItems = new ObservableCollection<Models.TodoItem>();

    public ObservableCollection<Models.TodoItem> AllItems { get { return this.allItems; } }

    public void AddTodoItem(ImageSource _image, double picSize, string title, string description, DateTimeOffset dateTime, bool? isChecked = false, int opacity = 0)
    {
        this.allItems.Add(new Models.TodoItem(_image, picSize, title, description, dateTime, isChecked, opacity));
    }

    private Models.TodoItem _selectedItem;

    public Models.TodoItem selectedItem
    {
        get { return _selectedItem; }
        set { this._selectedItem = value; }
    }

    public void RemoveTodoItem(string id)
    {
        //DIY

        //set selectedItem to null after remove
        //this.selectedItem = null;
        if (this.selectedItem != null)
        {
            AllItems.Remove(selectedItem);
            this.selectedItem = null;
        }
    }

    public void UpdateTodaItem(string id, ImageSource image, double picSize, string title, string description, DateTimeOffset date, bool? isChecked, int opacity)
    {
        //DIY

        //set selectedItem to null after remove
        //this.selectedItem = null;
        if (this.selectedItem != null)
        {
            if (selectedItem != null) this.selectedItem.UpdateItem(image, picSize, title, description, date, isChecked, opacity);
            this.selectedItem = null;
        }
    }
}
```

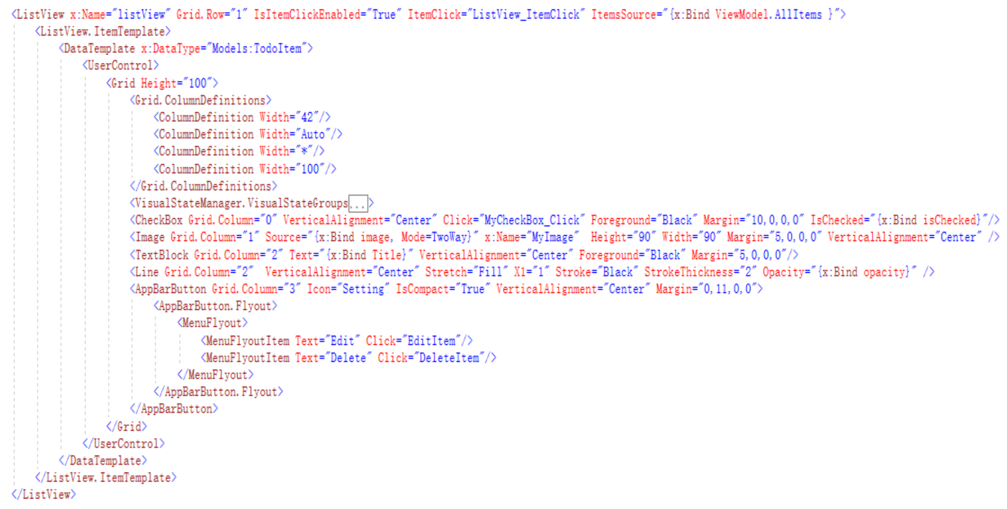
MainPage:

创建一个全局的静态 allItem，用来存储和删除 Item



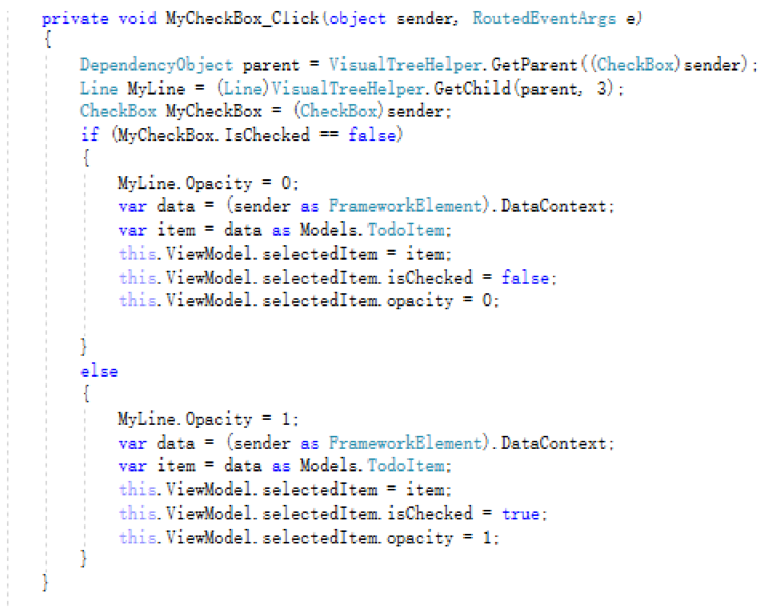
```
public static ViewModels.TodoItemViewModel allItem = new ViewModels.TodoItemViewModel();

// strongly-typed view models enable x:bind
public ViewModels.TodoItemViewModel ViewModel { get { return allItem; } }
```

将 Item 模块化并且绑定



```
<ListView x:Name="listView" Grid.Row="1" IsItemClickEnabled="True" ItemClick="ListView_ItemClick" ItemsSource="{x:Bind ViewModel.AllItems }">
    <ListView.ItemTemplate>
        <DataTemplate x:DataType="Models:TodoItem">
            <UserControl>
                <Grid Height="100">
                    <Grid.ColumnDefinitions>
                        <ColumnDefinition Width="42"/>
                        <ColumnDefinition Width="Auto"/>
                        <ColumnDefinition Width="*"/>
                        <ColumnDefinition Width="100"/>
                    </Grid.ColumnDefinitions>
                    <VisualStateManager.VisualStateGroups...>
                    <CheckBox Grid.Column="0" VerticalAlignment="Center" Click="MyCheckBox_Click" Foreground="Black" Margin="10,0,0,0" IsChecked="{x:Bind isChecked}"/>
                    <Image Grid.Column="1" Source="{x:Bind image, Mode=TwoWay}" x:Name="MyImage"  Height="90" Width="90" Margin="5,0,0,0" VerticalAlignment="Center" />
                    <TextBlock Grid.Column="2" Text="{x:Bind Title}" VerticalAlignment="Center" Foreground="Black" Margin="5,0,0,0"/>
                    <Line Grid.Column="2"  VerticalAlignment="Center" Stretch="Fill" X1="1" Stroke="Black" StrokeThickness="2" Opacity="{x:Bind opacity}" />
                    <AppBarButton Grid.Column="3" Icon="Setting" IsCompact="True" VerticalAlignment="Center" Margin="0,11,0,0">
                        <AppBarButton.Flyout>
                            <MenuFlyout>
                                <MenuFlyoutItem Text="Edit" Click="EditItem"/>
                                <MenuFlyoutItem Text="Delete" Click="DeleteItem"/>
                            </MenuFlyout>
                        </AppBarButton.Flyout>
                    </AppBarButton>
                </Grid>
            </UserControl>
        </DataTemplate>
    </ListView.ItemTemplate>
</ListView>
```

Item 的 check 部分：



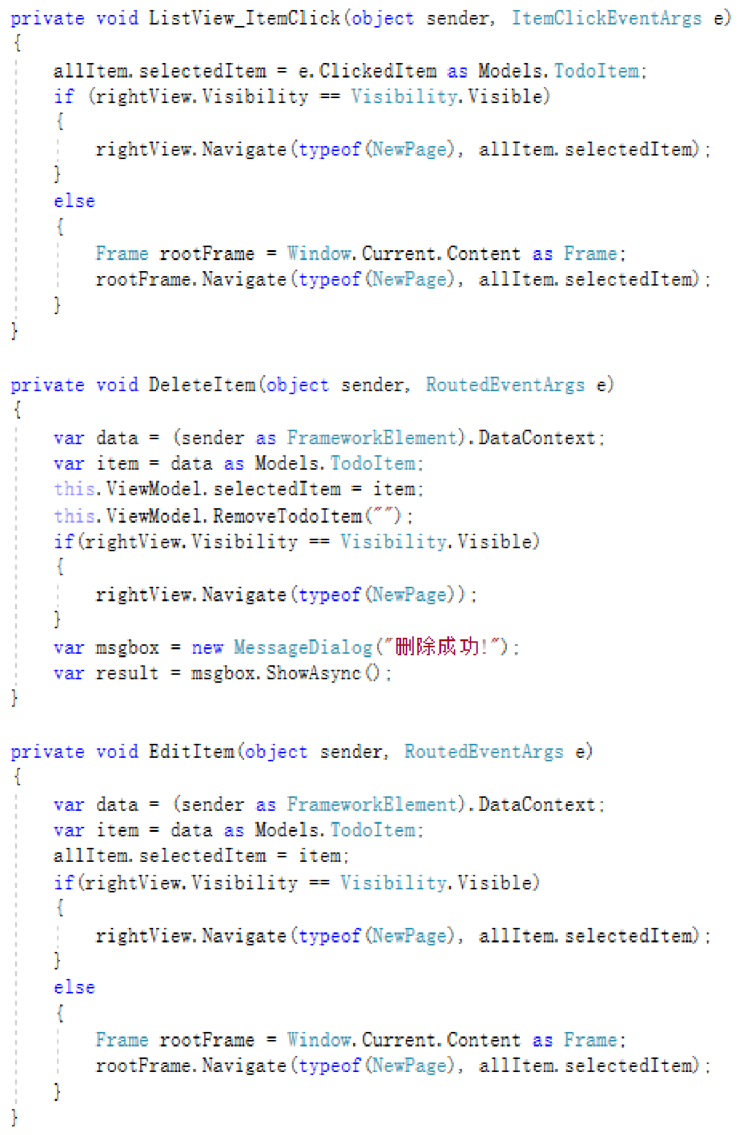
```
private void MyCheckBox_Click(object sender, RoutedEventArgs e)
{
    DependencyObject parent = VisualTreeHelper.GetParent((CheckBox)sender);
    Line MyLine = (Line)VisualTreeHelper.GetChild(parent, 3);
    CheckBox MyCheckBox = (CheckBox)sender;
    if (MyCheckBox.IsChecked == false)
    {
        MyLine.Opacity = 0;
        var data = (sender as FrameworkElement).DataContext;
        var item = data as Models.TodoItem;
        this.ViewModel.selectedItem = item;
        this.ViewModel.selectedItem.isChecked = false;
        this.ViewModel.selectedItem.opacity = 0;

    }
    else
    {
        MyLine.Opacity = 1;
        var data = (sender as FrameworkElement).DataContext;
        var item = data as Models.TodoItem;
        this.ViewModel.selectedItem = item;
        this.ViewModel.selectedItem.isChecked = true;
        this.ViewModel.selectedItem.opacity = 1;
    }
}
```

对 Item 的操作：

① 点击显示详情

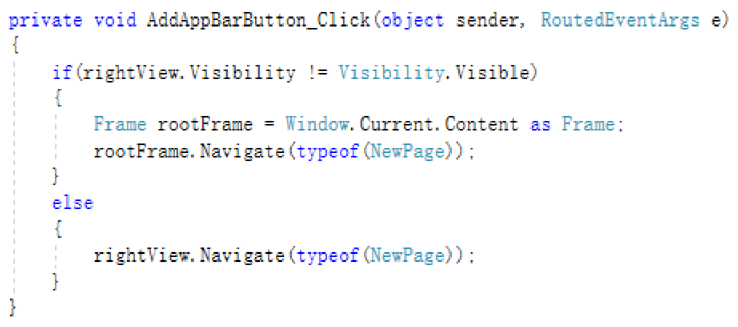
② 添加

③ 删除



```
private void ListView_ItemClick(object sender, ItemClickEventArgs e)
{
    allItem.selectedItem = e.ClickedItem as Models.TodoItem;
    if (rightView.Visibility == Visibility.Visible)
    {
        rightView.Navigate(typeof(NewPage), allItem.selectedItem);
    }
    else
    {
        Frame rootFrame = Window.Current.Content as Frame;
        rootFrame.Navigate(typeof(NewPage), allItem.selectedItem);
    }
}

private void DeleteItem(object sender, RoutedEventArgs e)
{
    var data = (sender as FrameworkElement).DataContext;
    var item = data as Models.TodoItem;
    this.ViewModel.selectedItem = item;
    this.ViewModel.RemoveTodoItem("");
    if(rightView.Visibility == Visibility.Visible)
    {
        rightView.Navigate(typeof(NewPage));
    }
    var msgbox = new MessageDialog("删除成功!");
    var result = msgbox.ShowAsync();
}

private void EditItem(object sender, RoutedEventArgs e)
{
    var data = (sender as FrameworkElement).DataContext;
    var item = data as Models.TodoItem;
    allItem.selectedItem = item;
    if(rightView.Visibility == Visibility.Visible)
    {
        rightView.Navigate(typeof(NewPage), allItem.selectedItem);
    }
    else
    {
        Frame rootFrame = Window.Current.Content as Frame;
        rootFrame.Navigate(typeof(NewPage), allItem.selectedItem);
    }
}
```

窄屏 “+” 按钮操作

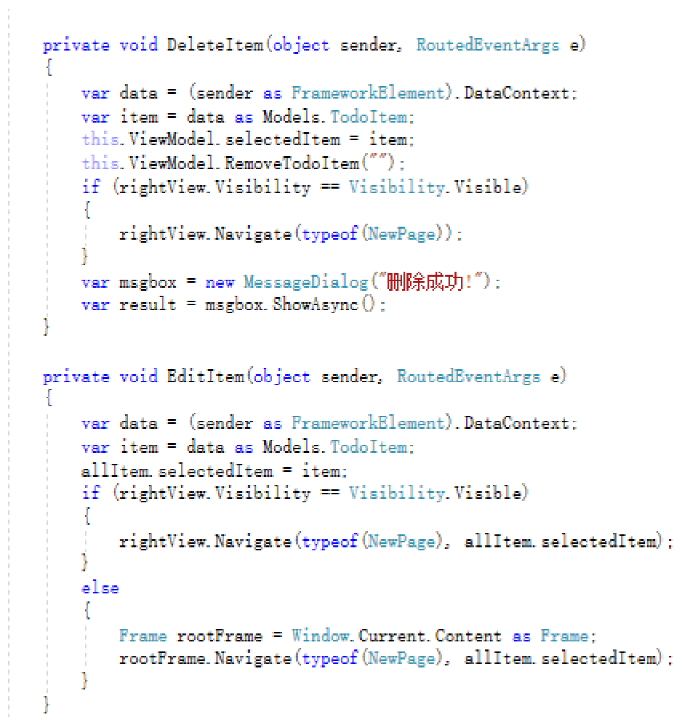


```
private void AddAppBarButton_Click(object sender, RoutedEventArgs e)
{
    if(rightView.Visibility != Visibility.Visible)
    {
        Frame rootFrame = Window.Current.Content as Frame;
        rootFrame.Navigate(typeof(NewPage));
    }
    else
    {
        rightView.Navigate(typeof(NewPage));
    }
}
```

齿轮操作：

① 修改

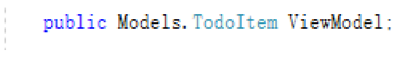
② 删除



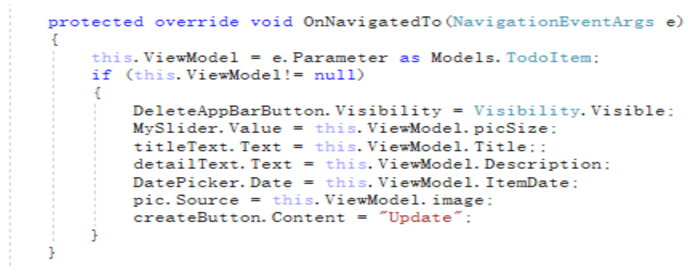
```
private void DeleteItem(object sender, RoutedEventArgs e)
{
    var data = (sender as FrameworkElement).DataContext;
    var item = data as Models.TodoItem;
    this.ViewModel.selectedItem = item;
    this.ViewModel.RemoveTodoItem("");
    if (rightView.Visibility == Visibility.Visible)
    {
        rightView.Navigate(typeof(NewPage));
    }
    var msgbox = new MessageDialog("删除成功!");
    var result = msgbox.ShowAsync();
}

private void EditItem(object sender, RoutedEventArgs e)
{
    var data = (sender as FrameworkElement).DataContext;
    var item = data as Models.TodoItem;
    allItem.selectedItem = item;
    if (rightView.Visibility == Visibility.Visible)
    {
        rightView.Navigate(typeof(NewPage), allItem.selectedItem);
    }
    else
    {
        Frame rootFrame = Window.Current.Content as Frame;
        rootFrame.Navigate(typeof(NewPage), allItem.selectedItem);
    }
}
```

NewPage:

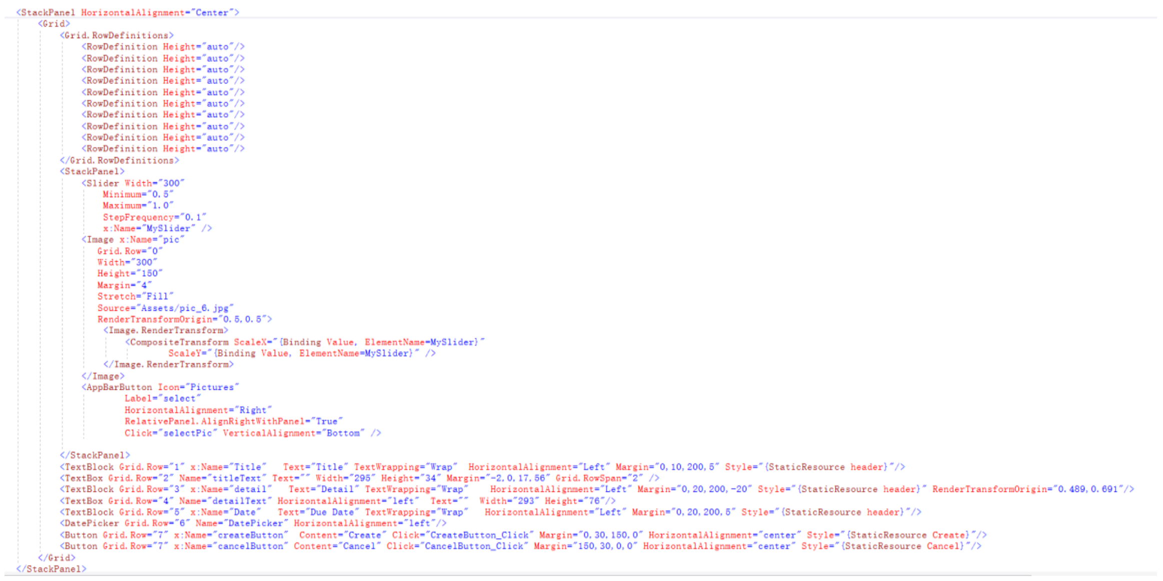
创建一个 Item，接收 MainPage 传来的 Item 的详情数据



```
public Models.TodoItem ViewModel;
```



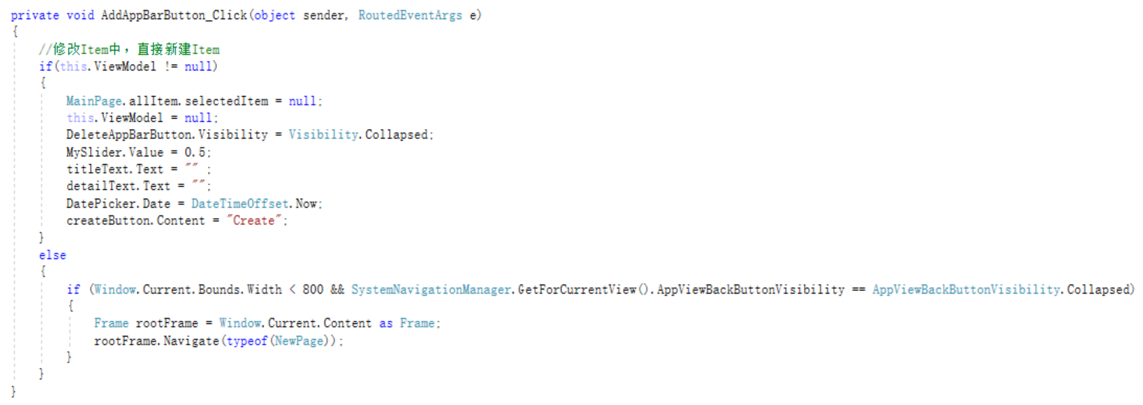
```
protected override void OnNavigatedTo(NavigationEventArgs e)
{
    this.ViewModel = e.Parameter as Models.TodoItem;
    if (this.ViewModel!= null)
    {
        DeleteAppBarButton.Visibility = Visibility.Visible;
        MySlider.Value = this.ViewModel.picSize;
        titleText.Text = this.ViewModel.Title;;
        detailText.Text = this.ViewModel.Description;
        DatePicker.Date = this.ViewModel.ItemDate;
        pic.Source = this.ViewModel.image;
        createButton.Content = "Update";
    }
}
```



```
<StackPanel HorizontalAlignment="Center">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="auto"/>
            <RowDefinition Height="auto"/>
            <RowDefinition Height="auto"/>
            <RowDefinition Height="auto"/>
            <RowDefinition Height="auto"/>
            <RowDefinition Height="auto"/>
            <RowDefinition Height="auto"/>
            <RowDefinition Height="auto"/>
            <RowDefinition Height="auto"/>
            <RowDefinition Height="auto"/>
        </Grid.RowDefinitions>
        <StackPanel>
            <Slider Width="300"
                Minimum="0.5"
                Maximum="1.0"
                StepFrequency="0.1"
                x:Name="MySlider" />
            <Image x:Name="pic"
               Grid.Row="0"
               Width="300"
               Height="150"
               Margin="4"
               Stretch="Fill"
               Source="Assets/pic_6.jpg"
               RenderTransformOrigin="0.5,0.5">
                <Image.RenderTransform>
                    <CompositeTransform ScaleX="{Binding Value, ElementName=MySlider}"
                            ScaleY="{Binding Value, ElementName=MySlider}" />
                </Image.RenderTransform>
            </Image>
            <AppBarButton Icon="Pictures"
                    Label="select"
                    HorizontalAlignment="Right"
                    RelativePanel.AlignRightWithPanel="True"
                    Click="selectPic" VerticalAlignment="Bottom" />

        </StackPanel>
        <TextBlock Grid.Row="1" x:Name="Title"   Text="Title" TextWrapping="Wrap"  HorizontalAlignment="Left" Margin="0,10,200,5" Style="{StaticResource header}"/>
        <TextBox Grid.Row="2" Name="titleText" Text="" Width="295" Height="34" Margin="-2,0,17,56" Grid.RowSpan="2" />
        <TextBlock Grid.Row="3" x:Name="detail"   Text="Detail" TextWrapping="Wrap"    HorizontalAlignment="Left" Margin="0,20,200,-20" Style="{StaticResource header}" RenderTransformOrigin="0.489,0.691"/>
        <TextBox Grid.Row="4" Name="detailText" HorizontalAlignment="left"  Text=""  Width="293" Height="76"/>
        <TextBlock Grid.Row="5" x:Name="Date"   Text="Due Date" TextWrapping="Wrap"   HorizontalAlignment="Left" Margin="0,20,200,5" Style="{StaticResource header}"/>
        <DatePicker Grid.Row="6" Name="DatePicker" HorizontalAlignment="left"/>
        <Button Grid.Row="7" x:Name="createButton"  Content="Create" Click="CreateButton_Click" Margin="0,30,150,0" HorizontalAlignment="center" Style="{StaticResource Create}"/>
        <Button Grid.Row="7" x:Name="cancelButton" Content="Cancel" Click="CancelButton_Click" Margin="150,30,0,0" HorizontalAlignment="center" Style="{StaticResource Cancel}"/>
    </Grid>
</StackPanel>
```

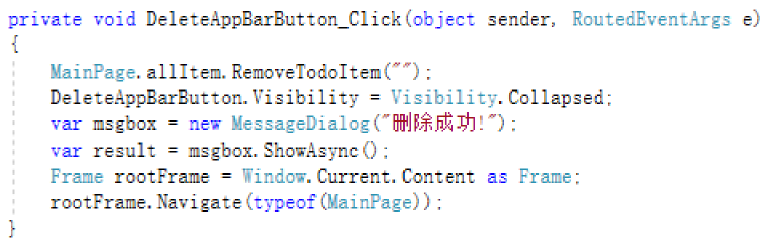
按钮 ‘+’ 操作

正在修改Item时候,点击直接到一个新的页面(个人感觉这样比较方便使用所以这样)



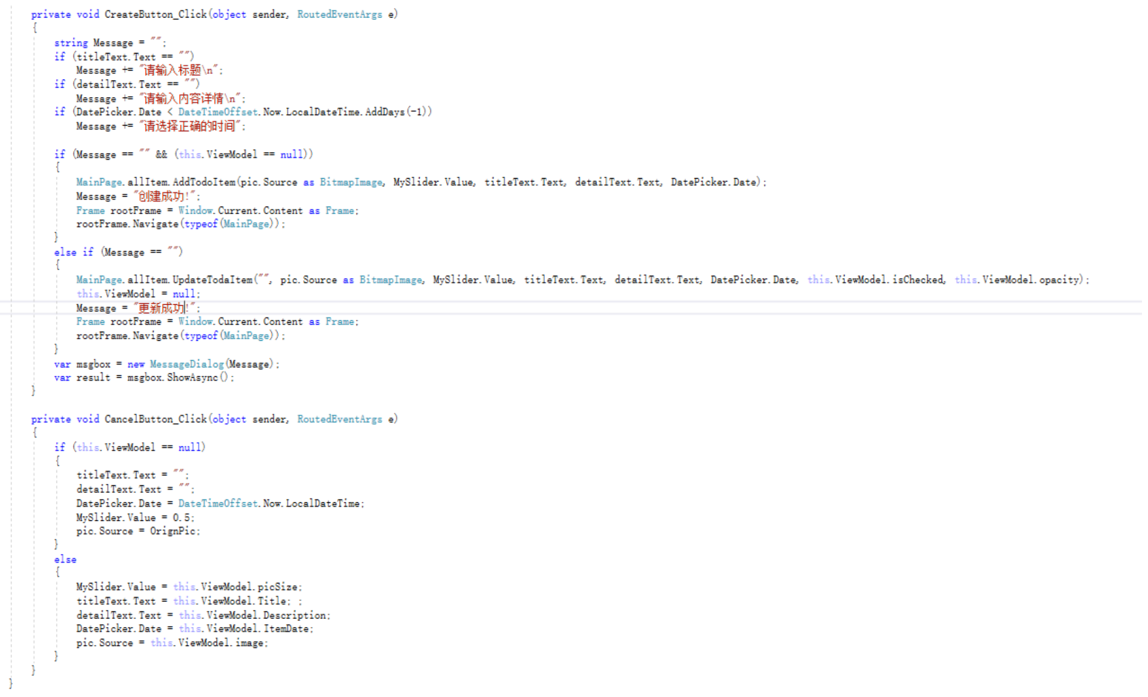
```
private void AddAppBarButton_Click(object sender, RoutedEventArgs e)
{
    //修改Item中，直接新建Item
    if(this.ViewModel != null)
    {
        MainPage.allItem.selectedItem = null;
        this.ViewModel = null;
        DeleteAppBarButton.Visibility = Visibility.Collapsed;
        MySlider.Value = 0.5;
        titleText.Text = "" ;
        detailText.Text = "";
        DatePicker.Date = DateTimeOffset.Now;
        createButton.Content = "Create";
    }
    else
    {
        if (Window.Current.Bounds.Width < 800 && SystemNavigationManager.GetForCurrentView().AppViewBackButtonVisibility == AppViewBackButtonVisibility.Collapsed)
        {
            Frame rootFrame = Window.Current.Content as Frame;
            rootFrame.Navigate(typeof(NewPage));
        }
    }
}
```

删除操作



```
private void DeleteAppBarButton_Click(object sender, RoutedEventArgs e)
{
    MainPage.allItem.RemoveTodoItem("");
    DeleteAppBarButton.Visibility = Visibility.Collapsed;
    var msgbox = new MessageDialog("删除成功!");
    var result = msgbox.ShowAsync();
    Frame rootFrame = Window.Current.Content as Frame;
    rootFrame.Navigate(typeof(MainPage));
}
```

创建 更新 取消 按钮逻辑操作



```
private void CreateButton_Click(object sender, RoutedEventArgs e)
{
    string Message = "";
    if (titleText.Text == "")
        Message += "请输入标题\n";
    if (detailText.Text == "")
        Message += "请输入内容详情\n";
    if (DatePicker.Date < DateTimeOffset.Now.LocalDateTime.AddDays(-1))
        Message += "请选择正确的时间";

    if (Message == "" && (this.ViewModel == null))
    {
        MainPage.allItem.AddTodoItem(pic.Source as BitmapImage, MySlider.Value, titleText.Text, detailText.Text, DatePicker.Date);
        Message = "创建成功!";
        Frame rootFrame = Window.Current.Content as Frame;
        rootFrame.Navigate(typeof(MainPage));
    }
    else if (Message == "")
    {
        MainPage.allItem.UpdateTodaItem("", pic.Source as BitmapImage, MySlider.Value, titleText.Text, detailText.Text, DatePicker.Date, this.ViewModel.isChecked, this.ViewModel.opacity);
        this.ViewModel = null;
        Message = "更新成功!";
        Frame rootFrame = Window.Current.Content as Frame;
        rootFrame.Navigate(typeof(MainPage));
    }
    var msgbox = new MessageDialog(Message);
    var result = msgbox.ShowAsync();
}

private void CancelButton_Click(object sender, RoutedEventArgs e)
{
    if (this.ViewModel == null)
    {
        titleText.Text = "";
        detailText.Text = "";
        DatePicker.Date = DateTimeOffset.Now.LocalDateTime;
        MySlider.Value = 0.5;
        pic.Source = OrignPic;
    }
    else
    {
        MySlider.Value = this.ViewModel.picSize;
        titleText.Text = this.ViewModel.Title; ;
        detailText.Text = this.ViewModel.Description;
        DatePicker.Date = this.ViewModel.ItemDate;
        pic.Source = this.ViewModel.image;
    }
}
}
```

四、亮点与改进（可选）

齿轮可以操作 Item 的修改和删除

① 逻辑部分处理的不够规范，有点混乱

② 很多 C#的操作都不清楚，代码有点冗余

③ 页面跳转的逻辑感觉现在写的不是很清晰，方法有点暴力

五、遇到的问题

① MainPage 怎么显示两个页面

② 页面怎么自适应

③ 数据绑定

④ 两个页面之间的数据绑定

六、思考与总结

这次实验问题很多，很多东西都不知道，只能不断翻博客，查文档，还有问别人，对 xmal 和 c#没有系统基础学习，平时有时间应该自己主动学习下。MVVM 模式不熟悉，使用起来不是很规范。