**Contoso Jobs**

**Lab 3. Removing Modality**

In this Lab you will change the way a Job is created in order that it doesn’t block the rest of the App from being used by other users.

The main modal operations are creation and editing of the jobs. It would be better if jobs on the screen were directly editable.

Defining a new job would create a new floating job control that you can edit in place.

This Lab continues from the previous Lab, or you can start from the start point in the folder named *3. Start Contoso Jobs Modality*

1. Open the existing **Contoso Jobs.sln** solution file in **Visual Studio 2015**
2. Open the **JobControl.xaml** file in the **Views** folder. Change the **TextBlock** controls to **TextBox** controls and make the Binding Mode **TwoWay**

If you haven’t already done this in a previous exercise, set the **InkCanvas** property **IsHitTestVisible** to False so that a finger touching it doesn’t register, only an active stylus will interact with the **InkCanvas**.

…

<TextBox x:Name="TitleTextBlock" TextWrapping="Wrap"

Text="{Binding Title, Mode=TwoWay}"

FontSize="{StaticResource TitleFontSize}"/>

<Grid Grid.Row="1">

<TextBox x:Name="DescriptionTextBlock" Margin="0" TextWrapping="Wrap"

Text="{Binding Description, Mode=TwoWay}" VerticalAlignment="Center"

Height="150"

FontSize="{StaticResource TextFontSize}"/>

<InkCanvas x:Name="Annotation" Height="150"

IsHitTestVisible="False" />

<Ellipse Width="20" Height="20" VerticalAlignment="Top"

HorizontalAlignment="Right" Fill="Black"

Margin="55,0" PointerPressed="Ellipse\_PointerPressed" />

<Ellipse Width="20" Height="20" VerticalAlignment="Top"

HorizontalAlignment="Right" Fill="Blue"

Margin="30,0" PointerPressed="Ellipse\_PointerPressed" />

<Ellipse Width="20" Height="20" VerticalAlignment="Top"

HorizontalAlignment="Right" Fill="Red"

Margin="5, 0" PointerPressed="Ellipse\_PointerPressed" />

</Grid>

…

1. Build and Run that App (F5). You can now edit the description and title in place as well as the Ink.

It is now harder to pick up and move the job. You need a ‘handle’ to grab the Job and movie it.

1. In the JobControl.xaml file add a handle to the top of the JobControl and move the TitleTextBox down to make space for the handle.

<Grid Margin="5">

<Grid.RowDefinitions>

<RowDefinition Height=".4\*"/>

<RowDefinition Height=".3\*"/>

<RowDefinition Height=".3\*" />

</Grid.RowDefinitions>

<Border HorizontalAlignment="Stretch" Height="15"

Background="Transparent" VerticalAlignment="Top">

<Grid VerticalAlignment="Top">

<Rectangle HorizontalAlignment="Stretch" Fill="Black" Height="2"

Margin="2,2,2,0" VerticalAlignment="Top" />

<Rectangle HorizontalAlignment="Stretch" Fill="Black" Height="2"

Margin="2,6,2,0" VerticalAlignment="Top"/>

<Rectangle HorizontalAlignment="Stretch" Fill="Black" Height="2"

Margin="2,10,2,0" VerticalAlignment="Top"/>

</Grid>

</Border>

<TextBox Name="TitleTextBox" Margin="0,15,0,0" Text="{Binding Title, Mode=TwoWay}"

BorderThickness="0"

FontSize="{StaticResource TitleFontSize}"/>

1. As you drag a JobControl around it doesn’t maintain its visual appearance. To fix that modify the RowDefinitions to have fixed height rows.

<Grid.RowDefinitions>

<RowDefinition Height="50"/>

<RowDefinition Height="150"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

1. The complete JobControl XAML should now look like this:

<UserControl

x:Class="Contoso\_Jobs.Views.JobControl"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:local="using:Contoso\_Jobs.Views"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d"

>

<Border x:Name="Border" BorderBrush="{Binding Parent.Highlight, ElementName=Border}"

Background="{Binding Parent.Fill, ElementName=Border}" BorderThickness="1"

Margin="0,2" >

<Grid Margin="5" VerticalAlignment="Top">

<Grid.RowDefinitions>

<RowDefinition Height="50"/>

<RowDefinition Height="150"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Border HorizontalAlignment="Stretch" Height="15" Background="Transparent"

VerticalAlignment="Top">

<Grid VerticalAlignment="Top">

<Rectangle HorizontalAlignment="Stretch" Fill="Black" Height="2"

Margin="2,2,2,0" VerticalAlignment="Top" />

<Rectangle HorizontalAlignment="Stretch" Fill="Black" Height="2"

Margin="2,6,2,0" VerticalAlignment="Top"/>

<Rectangle HorizontalAlignment="Stretch" Fill="Black" Height="2"

Margin="2,10,2,0" VerticalAlignment="Top"/>

</Grid>

</Border>

<TextBox Name="TitleTextBox" Margin="0,15,0,0" Text="{Binding Title,

Mode=TwoWay}" BorderThickness="0"

FontSize="{StaticResource TitleFontSize}"/>

<Grid Grid.Row="1" VerticalAlignment="Top" >

<TextBox Name="DescriptionTextBox" Margin="0" TextWrapping="Wrap"

BorderThickness="0"

Text="{Binding Description, Mode=TwoWay}" Height="150"

FontSize="{StaticResource TextFontSize}"/>

<InkCanvas x:Name="Annotation" Height="150" IsHitTestVisible="False" />

<Ellipse Width="20" Height="20" VerticalAlignment="Top"

HorizontalAlignment="Right" Fill="Black"

Margin="55,0" PointerPressed="Ellipse\_PointerPressed" />

<Ellipse Width="20" Height="20" VerticalAlignment="Top"

HorizontalAlignment="Right" Fill="Blue"

Margin="30,0" PointerPressed="Ellipse\_PointerPressed" />

<Ellipse Width="20" Height="20" VerticalAlignment="Top"

HorizontalAlignment="Right" Fill="Red"

Margin="5, 0" PointerPressed="Ellipse\_PointerPressed" />

</Grid>

<Button Grid.Row="2" FontFamily="Segoe MDL2 Assets"

Content="{Binding Parent.ProgressButtonText, ElementName=Border}"

ToolTipService.ToolTip="Start work on this job"

Background="Transparent" Click="Progress\_Click" />

<Button Grid.Row="2" VerticalAlignment="Bottom" HorizontalAlignment="Right"

Background="Transparent" ToolTipService.ToolTip="Edit this job"

FontFamily="Segoe MDL2 Assets" Content="&#xE70F;" Click="Edit\_Click" />

</Grid>

</Border>

</UserControl>

1. Open The JobViewModel.cs file and modify the **CreateJob** method to return the Job it has created

internal Job CreateJob(string title, string description, List<InkStroke> strokes)

{

Job j = new Job() { Title = title, Description = description,

Status = JobStatus.Backlog, Strokes = strokes };

jobs.Add(j);

SaveJobs();

OnPropertyChanged("Backlog");

return j;

}

1. In the Jobs.xaml.cs file edit the **CreateJob\_Click** event handler to create a new floating Job control where the button is pushed

private async void CreateJob\_Click(object sender, RoutedEventArgs e)

{

Job j = jobsViewModel.CreateJob(string.Empty, string.Empty, new List<InkStroke>());

JobControl dragJobControl = new JobControl();

dragJobControl.DataContext = j;

//set the visual of the new dragging job control

dragJobControl.Fill = new SolidColorBrush(Colors.LightGray);

//position the new dragging job control on the grid in the

//same position as the original job control

dragJobControl.VerticalAlignment = VerticalAlignment.Top;

dragJobControl.HorizontalAlignment = HorizontalAlignment.Left;

dragJobControl.Width = this.ActualWidth / 3;

Button btn = sender as Button;

if (btn != null)

{

GeneralTransform gt = btn.TransformToVisual(JobsGrid);

TranslateTransform trans = new TranslateTransform();

Point p = gt.TransformPoint(new Point(0, 0));

trans.X = p.X;

trans.Y = p.Y;

dragJobControl.RenderTransform = trans;

}

//add the new jobcontrol on the jobsgrid

JobsGrid.Children.Add(dragJobControl);

//add a recognizer for the new dragging job control

GestureRecognizer recognizer = new GestureRecognizer();

ManipulationInputProcessor manipulationProcessor = new

ManipulationInputProcessor(recognizer, dragJobControl, JobsGrid);

//manipulationProcessor.OnPointerPressed(sender, e);

//let the view model know that we are moving this job

jobsViewModel.MoveJob(j);

jobsViewModel.ShowingMenu = false;

}

You will need to add the Inking and Generic Collections namespaces to the top of the file

using Windows.UI.Input.Inking;  
using System.Collections.Generic;

1. Build and Run (**F5**) the App. When you create a new job it appears where the menu item is. You can create multiple new jobs without interrupting activity on other parts of the screen. It is no longer a modal operation.

The App needs to have a way for at least two jobs to be created at the same time. It is also important to remove the need for the HAMBURGER menu which will be hard to reach for some people and is stuck in the top left of the screen.

1. Open the Jobs.xaml file. In here you are going to add vertical bars, one down each side of the screen.

The bars will host buttons that interact with the application.

Add columns each side of the JobsGrid to support the button columns

<Grid Margin="10, 60,10,10" x:Name="JobsGrid" >

<Grid.ColumnDefinitions>

<ColumnDefinition Width="50"/>

<ColumnDefinition Width="0.3\*"/>

<ColumnDefinition Width="0.3\*"/>

<ColumnDefinition Width="0.3\*"/>

<ColumnDefinition Width="50"/>

</Grid.ColumnDefinitions>

1. Directly below the ColumnDefinitions add two Grid controls for the button columns

<Grid Background="Black" Grid.Column="0" >

<Button FontFamily="Segoe MDL2 Assets" Content="&#xE710;" Foreground="White"

Tag="Left" Width="50" Height="50" Background="Transparent"

Click="CreateJob\_Click" />

</Grid>

<Grid Background="Black" Grid.Column="4" >

<Button FontFamily="Segoe MDL2 Assets" Content="&#xE710;" Foreground="White"

Tag="Right" Width="50" Height="50" Background="Transparent"

Click="CreateJob\_Click" />

</Grid>

1. Modify the Grid.Column properties for the To Do, WIP, and Done columns, so they are shifted along by one, now that Grid.Column 0 is the left most button bar.

<StackPanel Grid.Column="1" Orientation="Vertical" Margin="5,0" BorderBrush="DarkGray"

BorderThickness="1">

<TextBlock Text="To Do" FontSize="32" Margin="10" />

<ListBox ItemsSource="{Binding Backlog}"

ItemTemplate="{StaticResource JobItemTemplate}">

<ListBox.ItemContainerStyle>

<Style TargetType="ListBoxItem">

<Setter Property="HorizontalContentAlignment" Value="Stretch" />

</Style>

</ListBox.ItemContainerStyle>

</ListBox>

</StackPanel>

<StackPanel Grid.Column="2" Orientation="Vertical" Margin="5,0" BorderBrush="DarkGray"

BorderThickness="1">

<TextBlock Text="W.I.P." FontSize="32" Margin="10" />

<ListBox ItemsSource="{Binding WIP}" ItemTemplate="{StaticResource JobItemTemplate}">

<ListBox.ItemContainerStyle>

<Style TargetType="ListBoxItem">

<Setter Property="HorizontalContentAlignment" Value="Stretch" />

</Style>

</ListBox.ItemContainerStyle>

</ListBox>

</StackPanel>

<StackPanel Grid.Column="3" Orientation="Vertical" Margin="5,0" BorderBrush="DarkGray"

BorderThickness="1">

<TextBlock Text="Done" FontSize="32" Margin="10" />

<ListBox ItemsSource="{Binding Done}"

ItemTemplate="{StaticResource JobItemTemplate}">

<ListBox.ItemContainerStyle>

<Style TargetType="ListBoxItem">

<Setter Property="HorizontalContentAlignment" Value="Stretch" />

</Style>

</ListBox.ItemContainerStyle>

</ListBox>

</StackPanel>

1. Return to the Jobs.xaml.cs file and ensure the newly created JobControl spans multiple columns. Also push the newly created control to appear on the right if the right hand button is pressed.

private void CreateJob\_Click(object sender, RoutedEventArgs e)

{

Job j = jobsViewModel.CreateJob(string.Empty, string.Empty, new List<InkStroke>());

JobControl dragJobControl = new JobControl();

dragJobControl.DataContext = j;

//set the visual of the new dragging job control

dragJobControl.Fill = new SolidColorBrush(Colors.LightGray);

//position the new dragging job control on the grid in the

//same position as the original job control

dragJobControl.VerticalAlignment = VerticalAlignment.Top;

dragJobControl.HorizontalAlignment = HorizontalAlignment.Left;

dragJobControl.Width = this.ActualWidth / 3;

Grid.SetColumnSpan(dragJobControl, 5);

Button btn = sender as Button;

if (btn != null)

{

GeneralTransform gt = btn.TransformToVisual(JobsGrid);

TranslateTransform trans = new TranslateTransform();

Point p = gt.TransformPoint(new Point(0, 0));

trans.X = p.X;

trans.Y = p.Y;

if (btn.Tag.ToString() == "Right")

{

trans.X = trans.X - dragJobControl.Width;

}

dragJobControl.RenderTransform = trans;

}

//add the new jobcontrol on the jobsgrid

JobsGrid.Children.Add(dragJobControl);

//add a recognizer for the new dragging job control

GestureRecognizer recognizer = new GestureRecognizer();

ManipulationInputProcessor manipulationProcessor =

new ManipulationInputProcessor(recognizer, dragJobControl, JobsGrid);

//let the view model know that we are moving this job

jobsViewModel.MoveJob(j);

jobsViewModel.ShowingMenu = false;

}

1. Build and Run the App (**F5**). You can now have two people create a new job at the same, one on each side of the screen. When a new Job is created, it does not interrupt the work of other people in other parts of the screen.

**Extra Activities**

Do you think the edit button is required anymore?

What could you do to remove the button or change the user interface so that the edit button is no longer required?