Revit 2022 - Abbreviation Helper

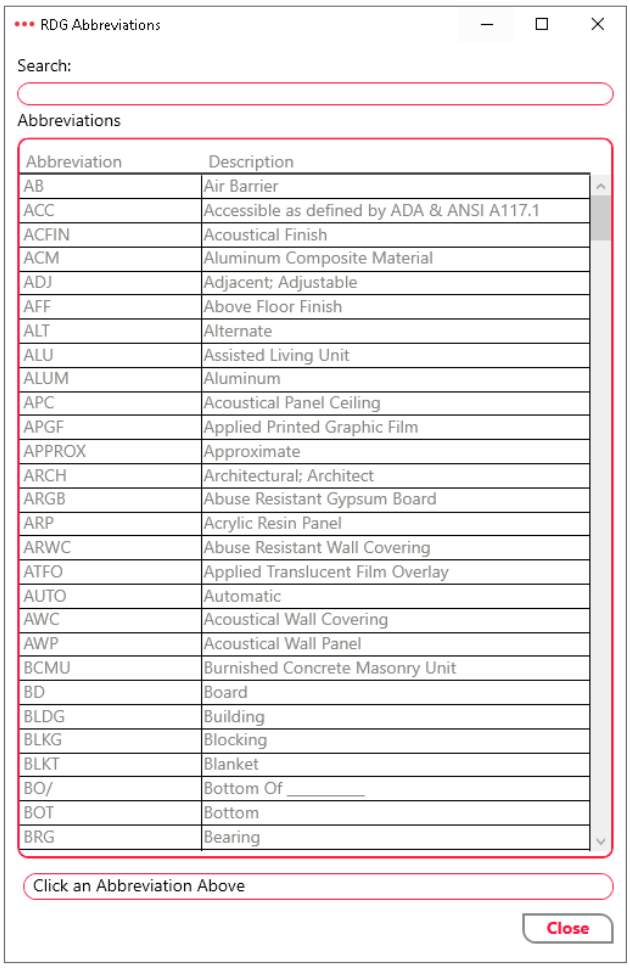
Example of Modeless Dialog for Abbreviation References

NOVEMBER 2021

This guide will walk through the steps that I used to provide user support for Abbreviation view schedule elements within a Revit 2022 model.

This process involves collecting all of the View Schedules in the model, then filtering to the correct one with a LINQ query and schedule Name. Once we have the correct schedule, another Filtered Element Collector is used to get all Elements in the Key Schedule and parse those for display in the form.

The following guide will also provide examples on the Windows Presentation Foundation (WPF) User Interface (UI) was developed and information connected using property binding using C# in Visual Studio 2019.



CONTACT:

Sean Page
Partner, Computational Designer
AIA, NCARB, LEED ap
https://rdgusa.com

References:
Building Coder: https://thebuildingcoder.typepad.com

Example of Modeless Dialog for Abbreviation References

NOVEMBER 2021

```
using System.Collections.Generic;
using System.Linq;
using Autodesk.Revit.Attributes;
using Autodesk.Revit.DB;
using Autodesk.Revit.UI;

namespace RDGRevit.Commands
{
    [Transaction(TransactionMode.Manual)]
    internal class Abbreviations : IExternalCommand
    {
        public Result Execute(ExternalCommandData commandData, ref string message, ElementSet elements)
        {
            //Get the current Document
            Document doc = commandData.Application.ActiveUIDocument.Document;
            //Custom Collection Class of Abbreviation Items
            List<WPF.Abbreviations> Abbreviations = new List<WPF.Abbreviations>();
            //Collect the Abbreviation Schedule. May be more useful here for a drop down / selection feature to prevent the
            wrong one being used
            ViewSchedule vs = new FilteredElementCollector(doc).OfCategory(BuiltInCategory.OST_Schedules)
                .Cast<ViewSchedule>().First(x => x.Name.Contains("ABBREVIATION"));
            //Get the Elements from the Schedule
            IList<Element> items = new FilteredElementCollector(doc, vs.Id).ToElements();
            //Create a string list to check again and remove from the items in the schedule (Notes)
            List<string> check = new List<string> { "1", "2", "3", "4", "5", "6", "7" };
            //Iterate each Element from the schedule, remove it if it matches the check list, and then get Parameter Values
            foreach(Element item in items)
            {
                if(!check.Contains(item.LookupParameter("RDG Abbreviations").AsString()))
                {
                    Abbreviations.Add(new WPF.Abbreviations()
                    {
                        Abbreviation = item.LookupParameter("RDG Abbreviations").AsString(),
                        Description = item.LookupParameter("RDG Complete Spelling").AsString()
                    });
                }
            }
            //Initialize the WPF Form and pass the Abbreviations List
            WPF.AbbreviationsWPF form = new WPF.AbbreviationsWPF(Abbreviations);
            //Magic Code from Jeremy Tammik to make the Modeless form know its Owner
            JtWindowHandle rvtwin = new JtWindowHandle(commandData.Application.MainWindowHandle);
            System.Windows.Interop.WindowInteropHelper helper = new System.Windows.Interop.WindowInteropHelper(form)
            {
                Owner = rvtwin.Handle
            };
            //Display the form Modeless
            form.Show();
            //Use Cancel here so no transaction is submitted
            return Result.Cancelled;
        }
    }
}
```

The above code is used for the initial ExternalCommand to collect the elements, parse the values, and then call the Form.

CONTACT:

Sean Page
Partner, Computational Designer
AIA, NCARB, LEED ap
https://rdgusa.com

Example of Modeless Dialog for Abbreviation References

NOVEMBER 2021

```
using System.Collections.Generic;
using System.Windows;

namespace RDGRevit.WPF
{
    /// <summary>
    /// Interaction logic for AbbreviationsWPF.xaml
    /// </summary>
    public partial class AbbreviationsWPF : Window
    {
        public AbbreviationsWPF(List<Abbreviations> _Abbreviations)
        {
            InitializeComponent();
            DataContext = new AbbreviationViewModel(_Abbreviations);
        }

        private void btnClose_Click(object sender, RoutedEventArgs e)
        {
            Close();
        }

        private void dgAbbreviations_SelectionChanged(object sender, System.Windows.Controls.SelectionChangedEventArgs e)
        {
            Abbreviations selected = (Abbreviations)dgAbbreviations.SelectedItem;
            Clipboard.SetText(selected.Abbreviation);
        }
    }
}
```

The above code is used for the code behind the WPF Form. Note that the DataContext is being set to minimize event handlers.

CONTACT:

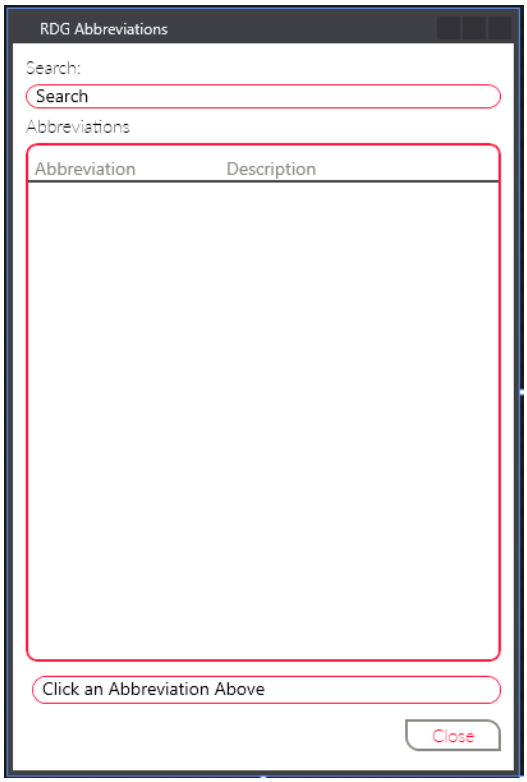
Sean Page
Partner, Computational Designer
AIA, NCARB, LEED ap
https://rdgusa.com

Example of Modeless Dialog for Abbreviation References

NOVEMBER 2021

```
<Window x:Class="RDGRevit.WPF.AbbreviationsWPF"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:local="clr-namespace:RDGRevit.WPF" Title="RDG Abbreviations" WindowStartupLocation="CenterScreen"
    Icon="../Resources/rdg.ico" d:DataContext="{d:DesignInstance Type=local:AbbreviationViewModel}"
    mc:Ignorable="d"
        Height="600" Width="400">

  <Window.Resources>
    <ResourceDictionary>
      <ResourceDictionary.MergedDictionaries>
        <ResourceDictionary Source="Styles/Colors.xaml"/>
        <ResourceDictionary Source="Styles/Fonts.xaml"/>
        <ResourceDictionary Source="Styles/Text.xaml"/>
        <ResourceDictionary Source="Styles/Buttons.xaml"/>
        <ResourceDictionary Source="Styles/DataGrid.xaml"/>
      </ResourceDictionary.MergedDictionaries>
    </ResourceDictionary>
  </Window.Resources>
```



The above code is used for the WPF form primary information like Title, Size, Icon and reference Resources. The remainder of the Window code is on the next page.

CONTACT:

Sean Page
Partner, Computational Designer
AIA, NCARB, LEED ap
https://rdgusa.com

Example of Modeless Dialog for Abbreviation References

NOVEMBER 2021

```
<Border Padding="10">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition/>
            <RowDefinition Height="35"/>
            <RowDefinition Height="35"/>
        </Grid.RowDefinitions>
        <StackPanel Grid.Row="0" Orientation="Vertical">
            <TextBlock Text="Search:"/>
            <TextBox x:Name="txtSearch" Text="{Binding Search, UpdateSourceTrigger=PropertyChanged}" ToolTip="Search for
Abbreviations" Style="{StaticResource Text}"/>
            <TextBlock Text="Abbreviations"/>
        </StackPanel>
        <Border x:Name="BorderCut" Grid.Row="1" Background="{StaticResource ForegroundLightBrush}"
            CornerRadius="8" BorderThickness="0" Margin="15 5"/>
        <DataGrid Grid.Row="1" x:Name="dgAbbreviations" Style="{StaticResource DGrid}"
            CanUserAddRows="False" CanUserResizeRows="False" AutoGenerateColumns="False"
            ItemsSource="{Binding FilteredAbbreviations}" SelectedValuePath="{Binding Abbreviation}"
            SelectedItem="{Binding SelectedItem,Mode=TwoWay}" SelectionChanged="dgAbbreviations_SelectionChanged">
            <DataGrid.OpacityMask>
                <VisualBrush Visual="{Binding ElementName=BorderCut}"/>
            </DataGrid.OpacityMask>
            <DataGrid.Columns>
                <DataGridTextColumn Header="Abbreviation" Binding="{Binding Abbreviation}" IsReadOnly="True" Width=".2*"
                    CellStyle="{StaticResource DefaultCell}" MinWidth="150"/>
                <DataGridTextColumn Header="Description" Binding="{Binding Description}" IsReadOnly="True" Width=".6*"
                    CellStyle="{StaticResource DefaultCell}">
                    <DataGridTextColumn.ElementStyle>
                        <Style>
                            <Setter Property="TextBlock.TextWrapping" Value="Wrap"/>
                        </Style>
                    </DataGridTextColumn.ElementStyle>
                </DataGridTextColumn>
            </DataGrid.Columns>
        </DataGrid>
        <DockPanel Grid.Row="2" LastChildFill="True">
            <TextBox Name="txtAbbreviation" Height="22" IsReadOnly="True"
                Text="{Binding SelectedItem.Abbreviation,ElementName=dgAbbreviations,
                StringFormat='Copied to Clipboard: {0}',FallbackValue=Click an Abbreviation Above}"
                CharacterCasing="Upper" Margin="5 0 0 0" Style="{StaticResource Text}"
                ToolTip="Provide a name to Rename the selected Scheme Entry"/>
        </DockPanel>
        <Button x:Name="btnClose" Content="Close" Grid.Row="3" Grid.ColumnSpan="2" Width="75" HorizontalAlignment="Right"
Margin="5 5 0 5"
            Click="btnClose_Click" IsCancel="True" Style="{StaticResource RDGButton}"/>
    </Grid>
  </Border>
</Window>
```


CONTACT:

Sean Page
Partner, Computational Designer
AIA, NCARB, LEED ap
https://rdgusa.com

The above code is used for the remainder of the Window and demonstrates the Binding and Static Resources.

Example of Modeless Dialog for Abbreviation References

NOVEMBER 2021

```
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel;

namespace RDGRevit.WPF
{
	internal class AbbreviationViewModel : INotifyPropertyChanged
	{
		public ObservableCollection<Abbreviations> abbreviations { get; set; }
		public event PropertyChangedEventHandler PropertyChanged;
		public string abbreviation { get; set; }
		public ICollectionView FilteredAbbreviations =>
System.Windows.Data.CollectionViewSource.GetDefaultView(abbreviations);

		public AbbreviationViewModel(List<Abbreviations> _abbreviations)
		{
			abbreviations = new ObservableCollection<Abbreviations>(_abbreviations);
			FilteredAbbreviations.Filter = new System.Predicate<object>(a => Filter(a as Abbreviations));
		}

		private bool Filter(Abbreviations _abbv)
		{
			return Search == null
				|| _abbv.Abbreviation.IndexOf(Search, System.StringComparison.OrdinalIgnoreCase) != -1
				|| _abbv.Description.IndexOf(Search, System.StringComparison.OrdinalIgnoreCase) != -1;
		}

		private string search;

		public string Search
		{
			get { return search; }
			set
			{
				search = value;
				NotifyPropertyChanged("Search");
				FilteredAbbreviations.Refresh();
			}
		}

		private void NotifyPropertyChanged(string propertyName = "")
		{
			PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
		}
	}

	public class Abbreviations
	{
		public string Abbreviation { get; set; }
		public string Description { get; set; }
	}
}
```

The above code is used for the View Model, Observable Collection, and other Properties for binding. The INotifyPropertyChanged interface is being implemented for updating the information displayed on the Form.

CONTACT:

Sean Page
Partner, Computational Designer
AIA, NCARB, LEED ap
https://rdgusa.com