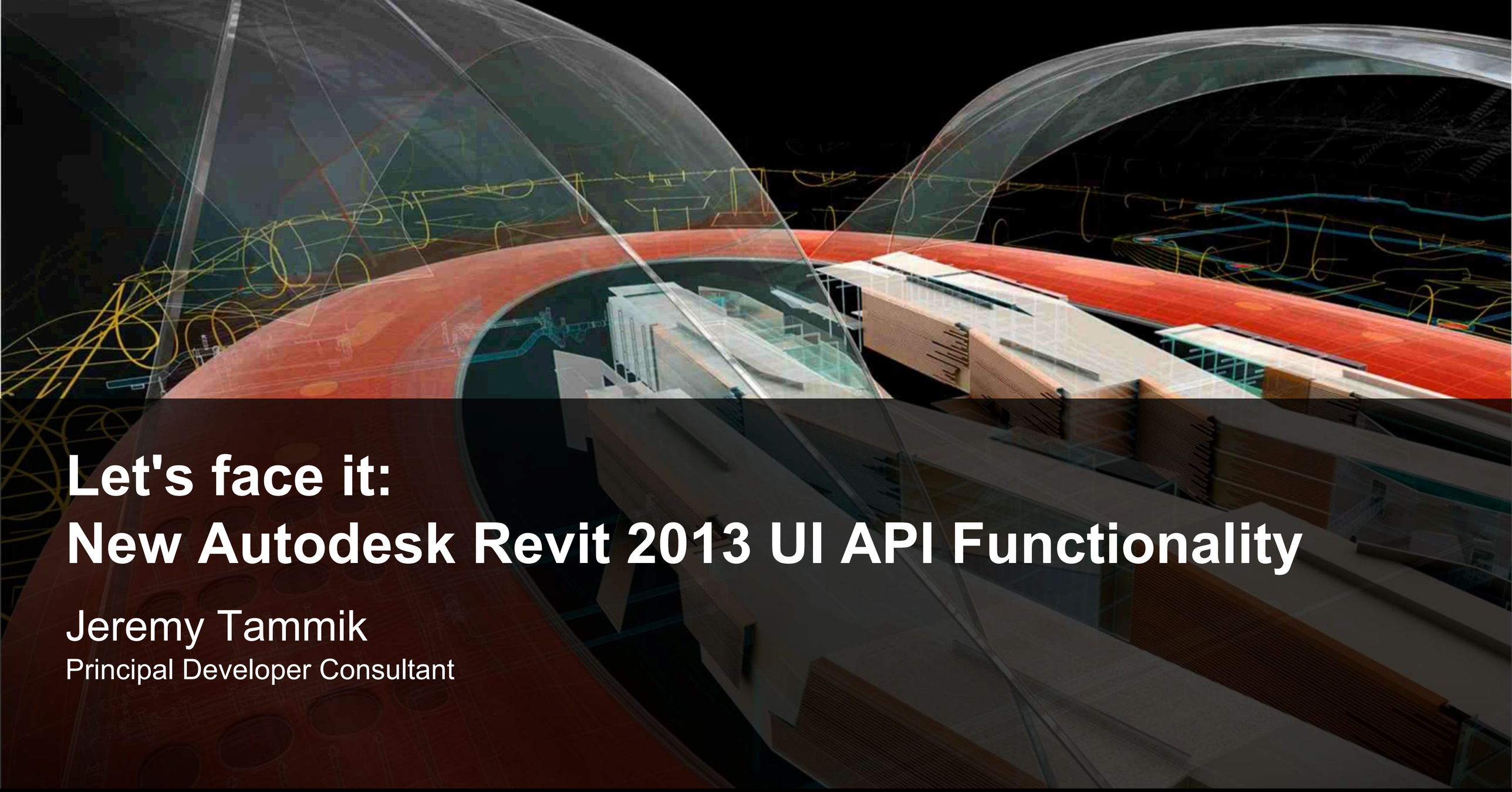


Let's face it:
New Autodesk Revit 2013 UI API Functionality

Jeremy Tammik
Principal Developer Consultant

About the Presenter

Jeremy Tammik
Principal Developer Consultant
Developer Technical Services
EMEA, Autodesk SARL

Jeremy is a member of the AEC workgroup of the Autodesk Developer Network ADN team, providing developer support, training, conference presentations, and blogging on the Revit API.

He joined Autodesk in 1988 as the technology evangelist responsible for European developer support to lecture, consult, and support AutoCAD application developers in Europe, the U.S., Australia, and Africa. He was a co-founder of ADGE, the AutoCAD Developer Group Europe, and a prolific author on AutoCAD application development. He left Autodesk in 1994 to work as an HVAC application developer, and then rejoined the company in 2005.

Jeremy graduated in mathematics and physics in Germany, worked as a teacher and translator, then as a C++ programmer on early GUI and multitasking projects. He is fluent in six European languages, vegetarian, has four kids, plays the flute, likes reading, travelling, theatre improvisation, yoga, carpentry, loves mountains, oceans, sports, dancing, and especially climbing.

Class Summary

- Revit 2013 API add-in integration functionality
- Continue Saikat's Revit UI API snapshot CP3272 which covers
  - Control of disciplines and add-in availability
  - Commands, keyboard shortcuts and the quick access toolbar
  - Replacing an existing Revit command
- Look in more depth at
  - Document management and View API
  - Progress bar notifications
  - Options dialogue WPF custom extensions
  - Embedding and controlling a Revit view
  - UIView and Windows device coordinates
  - Drag and drop
- Prerequisites
  - .NET programming and Revit API basics

Learning Objectives

- In depth understanding of Revit API add-in integration
- Add-in interaction using the new document management, view creation, settings, progress bar notification, WPF option dialogue extension, preview control, UIView and Windows coordinates, drag and drop, etc.
- Add-in alignment with the Revit user interface look and feel
- Know, understand and reuse Revit SDK and custom AU sample code

Agenda

- Document management and View API
- Revit progress bar notifications
- Options dialogue WPF custom extensions
- Embedding and controlling a Revit view
- Drag and drop
- UIView

Introduction

Autodesk Developer Network

www.autodesk.com/adnopen

www.autodesk.com/joinadn

- Access to almost all Autodesk software and SDK's
  - Includes early access to beta software
- Product direction through conferences
- Unlimited technical support
- API training classes
  - One to three free for professional members
- Marketing benefits
  - Exposure on autodesk.com
  - Promotional opportunities

Acronyms

- ADN Autodesk Developer Network
- AEC Architecture, Engineering, Construction
- API Application Programming Interface
- BIM Building Information Model
- GUI Graphical User Interface
- HVAC Heating, Ventilation, and Air Conditioning
- MEP Mechanical, Electrical, and Plumbing
- RAC Revit Architecture
- RME Revit MEP
- RST Revit Structure
- SDK Software Development Kit
- UI User Interface

Sample Applications

- Minimal simplified AU sample commands RevitUiApiNews
  - CmdAddOptionsTab
  - CmdDragDropApi
  - CmdPreviewControlSimple
  - CmdProgressWatcher
  - CmdUiView
- WinTooltip
  - Custom tooltip, UIView, Windows coordinates, Idling, ReferenceIntersector
- ProgressNotifier SDK sample
  - Subscribe to progress bar events
- UIAPI SDK sample
  - Add customised tabs to Options dialogue
  - Revit preview control
  - Drag and drop

Document Management and View API

Document Management and View API

- OpenAndActivateDocument enables opening a project file
- UIDocument.ActiveView enables view change within document
- UIView class provides access to UI views
  - Window location and Windows device coordinates
  - GetOpenUIViews lists all open views for a UI document
- View settings and creation
  - Steven Mycynek CP3133 Using the schedule and view APIs

View Settings

- View properties to read and write detail level, discipline, display style
  - ViewDetailLevel: coarse, medium, fine, etc.
  - ViewDiscipline: architectural, coordination, electrical, mechanical, plumbing, structural
  - DisplayStyle: Wireframe, HLR, Shading, ShadingWithEdges, Rendering, Realistic , FlatColors, RealisticWithEdges, Raytrace

View Creation

- View creation
  - Static Create methods on all View classes
  - Schedule, plan, 3D isometric and perspective
  - Specify view family type element id

Progress Bar Notifications

Progress Bar Notifications

- Subscribe to ProgressChanged event
- Event handler receives ProgressChangedEventArgs
  - Caption – progress bar caption describing operation in progress
  - Stage – current stage of the progress bar
    - Started, CaptionChanged, RangeChanged, PositionChanged, UserCancelled, Finished
  - LowerRange – lower limit of range, always zero
  - UpperRange – upper limit of range, any non-zero number
  - Position – value in [0, UpperRange] incremented with 'PositionChanged' stage

ProgressWatcher AU Sample

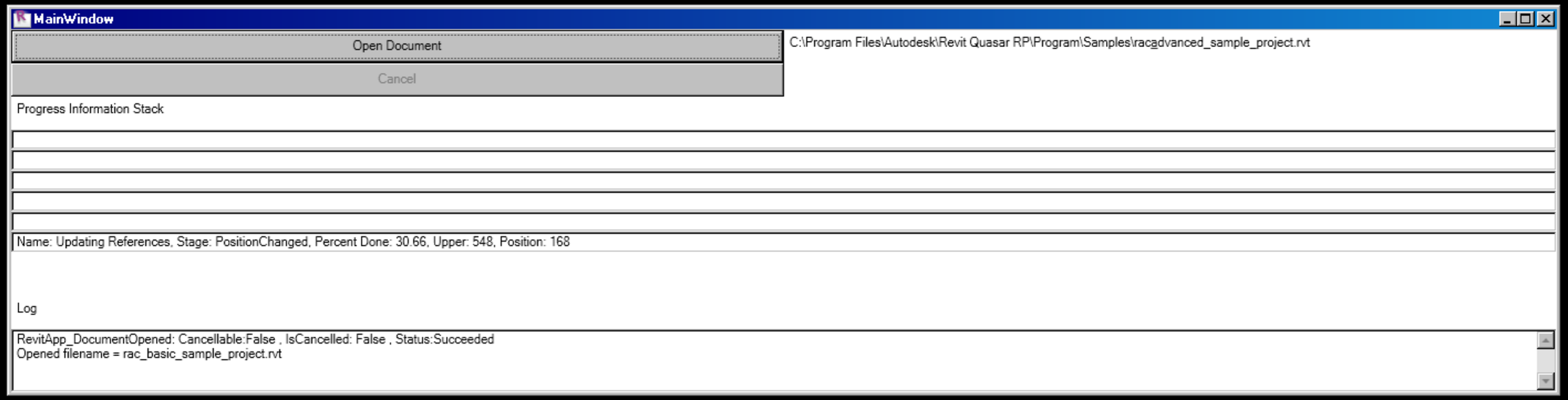
```
public Result Execute(
  ExternalCommandData commandData,
  ref string message,
  ElementSet elements )
{
  UIApplication uiapp = commandData.Application;
  Application app = uiapp.Application;

  app.ProgressChanged
    += new EventHandler<ProgressChangedEventArgs>(
      OnProgressChanged );

  return Result.Succeeded;
}

void OnProgressChanged(
  object sender,
  ProgressChangedEventArgs e )
{
  double percent = 100.0 * e.Position / e.UpperRange;

  Debug.Print( "'{0}' stage {1} position {2} [{3}, {4}] ({5}%)",
    e.Caption, e.Stage, e.Position, e.LowerRange, e.UpperRange,
    percent.ToString( "0.##" ) );
}
```

Sample Notifications Opening a Project

```
' ' stage Finished position 0 [0, 2147483647] (0%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage Started position 0 [0, 100] (0%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 0 [0, 100] (0%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 0 [0, 100] (0%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 0 [0, 100] (0%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 0 [0, 100] (0%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 0 [0, 100] (0%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 0 [0, 100] (0%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 0 [0, 100] (0%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 0 [0, 100] (0%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 0 [0, 100] (0%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 50 [0, 100] (50%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 50 [0, 100] (50%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 50 [0, 100] (50%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 75 [0, 100] (75%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 75 [0, 100] (75%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 50 [0, 100] (50%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 50 [0, 100] (50%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 50 [0, 100] (50%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 50 [0, 100] (50%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 75 [0, 100] (75%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 75 [0, 100] (75%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 75 [0, 100] (75%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage Finished position 75 [0, 100] (75%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage Started position 0 [0, 100] (0%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 0 [0, 100] (0%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 0 [0, 100] (0%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 0 [0, 100] (0%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 0 [0, 100] (0%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 0 [0, 100] (0%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 0 [0, 100] (0%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 0 [0, 100] (0%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 0 [0, 100] (0%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 25 [0, 100] (25%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 25 [0, 100] (25%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 25 [0, 100] (25%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage Finished position 25 [0, 100] (25%)
' ' stage Started position 0 [0, 2147483647] (0%)
' ' stage Finished position 0 [0, 2147483647] (0%)
' ' stage Started position 0 [0, 2147483647] (0%)
' ' stage Finished position 0 [0, 2147483647] (0%)
' ' stage Started position 0 [0, 2147483647] (0%)
' ' stage Finished position 0 [0, 2147483647] (0%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage Started position 0 [0, 100] (0%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 0 [0, 100] (0%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 0 [0, 100] (0%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 0 [0, 100] (0%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 0 [0, 100] (0%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 0 [0, 100] (0%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 0 [0, 100] (0%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 0 [0, 100] (0%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 0 [0, 100] (0%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 25 [0, 100] (25%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 25 [0, 100] (25%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage PositionChanged position 25 [0, 100] (25%)
'Drawing: floor_slab.rvt - 3D View: {3D}' stage Finished position 25 [0, 100] (25%)
```

ProgressNotifier SDK Sample

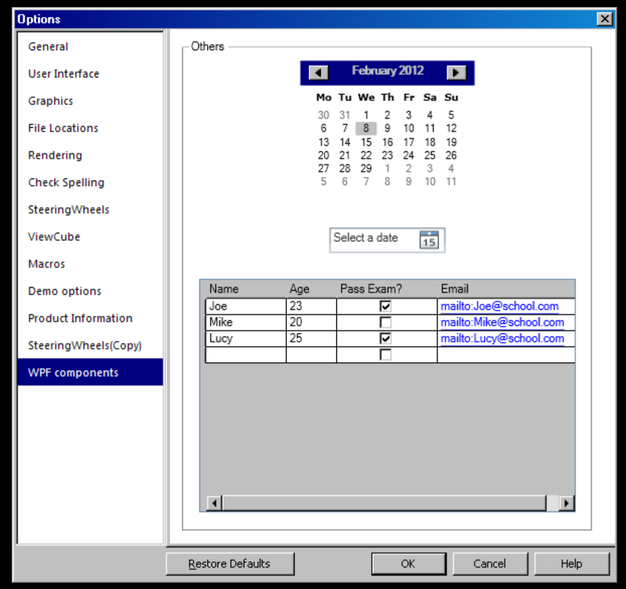
- Display progress information in a stack data structure
  - Subscribe to the ProgressChanged event
  - Access ProgressChangedEventArgs handler argument properties
  - Manage a stack of subtransaction progress information
  - Add-Ins > RvtSamples > Basics > ProgressNotifier > Open Document



Options Dialogue WPF Extensions

Options Dialogue Custom WPF Extensions

- Subscribe to UIApplication DisplayingOptionsDialog event
- Event handler receives DisplayingOptionsDialogEventArgs
- PagesCount property
  - Number of Options dialogue tabs including default Revit ones
- AddTab method
  - Add a tab with tab name and handler information provided by TabbedDialogExtension class instance



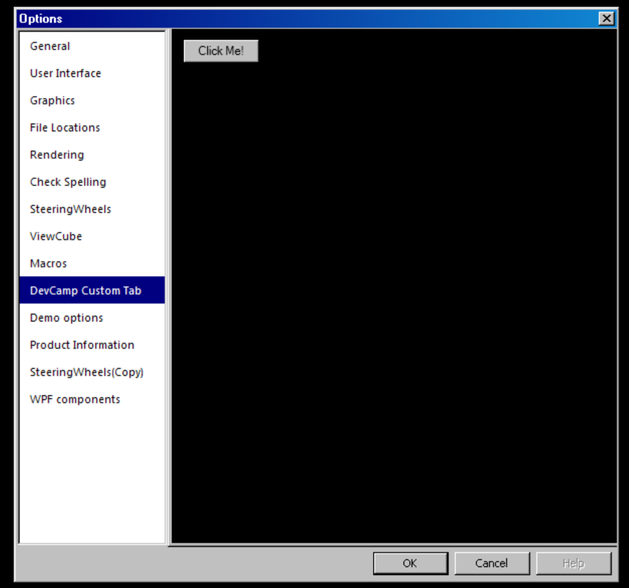
TabbedDialogExtension Class

- Constructor takes two arguments
  - WPF user control instance and OK handler
  - These cannot be changed
- Methods to get and set contextual help
- Properties to read
  - Control – the control
  - OnOKAction – the OK handler
- Properties to get and set
  - OnCancelAction – the cancel handler
  - OnRestoreDefaultsAction – the restore defaults handler

AU Sample Custom Tab

```
public partial class UserControl1 : UserControl
{
  string _name;

  public UserControl1( string name )
  {
    _name = name;

    InitializeComponent();
  }

  private void button1_Click(
    object sender,
    RoutedEventArgs e )
  {
    TaskDialog.Show( _name, "I was clicked..." );
  }

  public void OnOK()
  {
    TaskDialog.Show( _name, "OK" );
  }

  public void OnCancel()
  {
    TaskDialog.Show( _name, "OnCancel" );
  }

  public void OnRestoreDefaults()
  {
    TaskDialog.Show( _name, "OnRestoreDefaults" );
  }
}
```

```
public Result Execute(
  ExternalCommandData commandData,
  ref string message,
  ElementSet elements )
{
  UIApplication uiapp = commandData.Application;

  uiapp.DisplayingOptionsDialog
    += new EventHandler<
      DisplayingOptionsDialogEventArgs>(
      OnDisplayingOptionsDialog );

  return Result.Succeeded;
}

void OnDisplayingOptionsDialog(
  object sender,
  DisplayingOptionsDialogEventArgs e )
{
  UserControl1 c = new UserControl1(
    "DevCamp User Control" );

  e.AddTab( "DevCamp Custom Tab",
    new TabbedDialogExtension(
      c, c.OnOK ) );
}
```

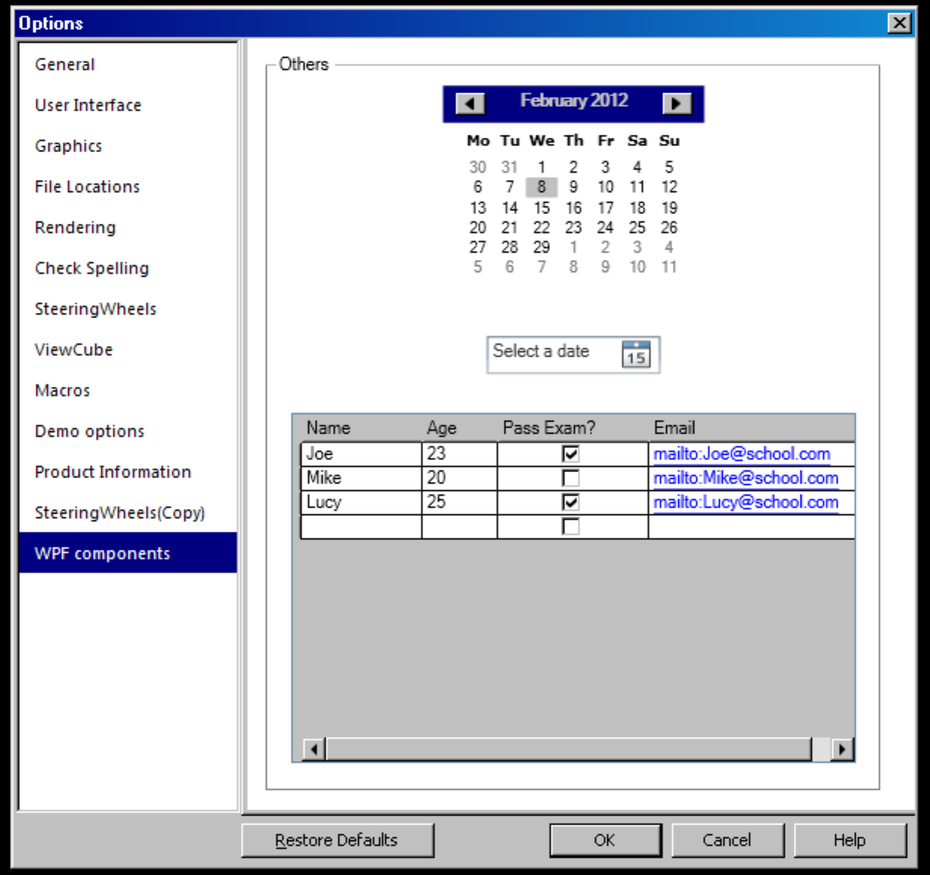


Additional References Required

- PresentationCore
- PresentationFramework
- System.Xaml
- WindowsBase

Revit SDK UIAPI Sample

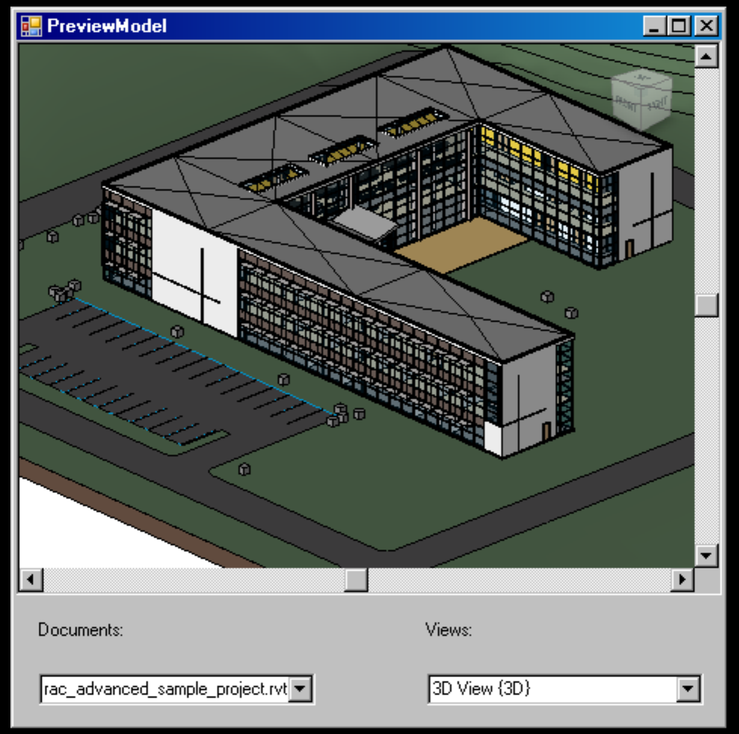
- Adds three tabs
- Demo options
- Product Information
- SteeringWheels(Copy)
- WPF components



Preview Control

Embed and Control a Revit View

- PreviewControl Class
- Enable browsing the Revit model
- Input document and view id
- Form or window host must be modal
- View can be any graphical view, i.e. printable
- View can be manipulated by view cube
- Visibility and graphical settings are effective



Using a Revit Preview Control

- Create a standard .NET form
- Insert WPF host

```
System.Windows.Forms.Integration.ElementHost
```

- Populate with Revit preview control

```
elementHost.Child = new PreviewControl( doc, view.Id );
```

- Dispose after use

```
PreviewControl vc = elementHost.Child
  as PreviewControl;

if( vc != null ) { vc.Dispose(); }
```

Determine Printable Views

```
IEnumerable<View> views
    = new FilteredElementCollector( doc )
      .OfClass( typeof( View ) )
      .Cast<View>()
      .Where<View>( v => v.CanBePrinted );
```

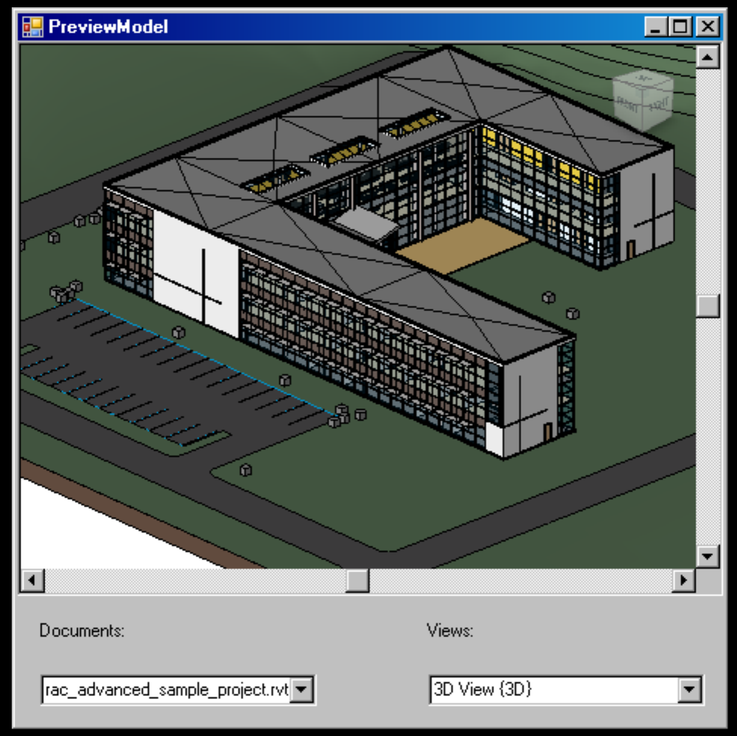
AU Sample PreviewControlSimple

```
using( PreviewControl pc
  = new PreviewControl( doc, view.Id ) )
{

#if CREATE_FORM_BY_CODE
  using( System.Windows.Forms.Form form
    = new System.Windows.Forms.Form() )
  {
    ElementHost elementHost = new ElementHost();

    elementHost.Location
      = new System.Drawing.Point( 0, 0 );

    elementHost.Dock = DockStyle.Fill;
    elementHost.TabIndex = 0;
    elementHost.Parent = form;
    elementHost.Child = pc;

    form.Text = _caption_prefix + view.Name;
    form.Controls.Add( elementHost );
    form.Size = new Size( 400, 400 );
    form.ShowDialog( owner );
  }
#else // if not CREATE_FORM_BY_CODE
  Form1 form = new Form1( pc );
  form.ShowDialog( owner );
#endif // CREATE_FORM_BY_CODE

}
```

```
IWin32Window revit_window
  = new JtWindowHandle(
    ComponentManager.ApplicationWindow );

UIApplication uiapp = commandData.Application;
UIDocument uidoc = uiapp.ActiveUIDocument;
Document doc = uidoc.Document;

View view = doc.ActiveView;

DisplayRevitView( doc, view,
  revit_window );
```

Revit SDK UIAPI PreviewControl Sample

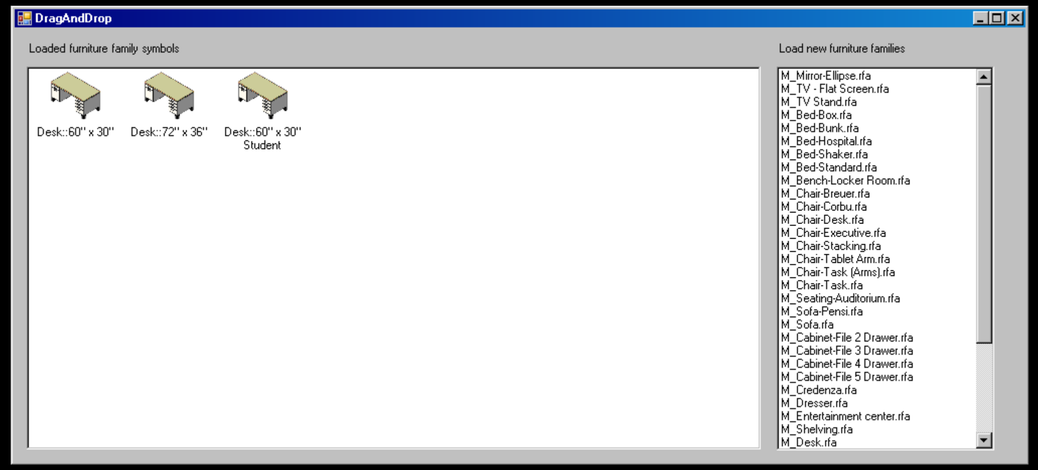
- Displays any view in any open document
- Includes perspective views
- Enables opening new documents



Drag and Drop

Drag and Drop API

- Two overloads of new static UIApplication DoDragDrop method
- DoDragDrop(ICollection<string>)
  - Initiate standard built-in Revit drag and drop operation
  - Collection of file names
- DoDragDrop(object, IDropHandler)
  - Initiate drag and drop operation with a custom drop implementation
- Designed for use in a modeless form
  - The one and only Revit API method not requiring valid Revit API context



Drag and Drop a List of Files

- One AutoCAD format or image file
  - A new import placement editor is started to import the file
- More than one AutoCAD format or image file
  - A new import placement editor is started for the first file
- One family file
  - The family is loaded, and an editor started to place an instance
- More than one family file
  - All the families are loaded
- More than one family file including other format files
  - Revit tries to open all the files
- **Valid** file or list of files: Revit uses them appropriately
- Any files are **not usable**: failure is signalled to user, but **no exception** is thrown and add-in is **not notified**

Initiating Standard Drag and Drop

```
private void listBox1_MouseMove(
  object sender,
  MouseEventArgs e )
{
  if( System.Windows.Forms.Control.MouseButtons
    == MouseButtons.Left )
  {
    FamilyListBoxMember member
      = (FamilyListBoxMember) listBox1.SelectedItem;

    // Use standard Revit drag and drop behavior

    List<String> data = new List<String>();

    data.Add( member.FullPath );

    UIApplication.DoDragDrop( data );
  }
}
```

AU Custom Drag and Drop Handler

```
public class LoadedFamilyDropHandler : IDropHandler
{
  public void Execute(
    UIDocument uidoc,
    object data )
  {
    ElementId familySymbolId = (ElementId) data;

    FamilySymbol symbol = uidoc.Document.GetElement(
      familySymbolId ) as FamilySymbol;

    if( symbol != null )
    {
      uidoc.PromptForFamilyInstancePlacement(
        symbol );
    }
  }
}
```

```
private void listView_MouseMove(
  object sender,
  MouseEventArgs e )
{
  if( Control.MouseButtons
    == MouseButtons.Left )
  {
    ListViewItem selectedItem
      = listView1.SelectedItems
        .Cast<ListViewItem>()
        .FirstOrDefault<ListViewItem>();

    if( selectedItem != null )
    {
      LoadedFamilyDropHandler myhandler
        = new LoadedFamilyDropHandler();

      UIApplication.DoDragDrop(
        selectedItem.Tag, myhandler );
    }
  }
}
```

AU Drag and Drop Sample

UIAPI Drag and Drop Sample

UIView and Windows Coordinates

UIView Methods

- GetWindowRectangle
  - Return view drawing area rectangle Windows device coordinates
- GetZoomCorners
  - Return view rectangle Revit model coordinates
- ZoomAndCenterRectangle
  - Zoom and centre view to a specified rectangle

UIView Sample Command

- Get the active view
- Get all UIView instances
- Determine the UIView for the active view
- Query and report its Windows and Revit coordinates
- Calculate new Revit coordinates to zoom in by 10%
- Call ZoomAndCenterRectangle to do so

UIView Sample Code

```
View view = doc.ActiveView;

UIView uiview = null;

IList<UIView> uiviews = uidoc.GetOpenUIViews();

foreach( UIView uv in uiviews )
{
  if( uv.ViewId.Equals( view.Id ) )
  {
    uiview = uv;
    break;
  }
}

Rectangle rect = uiview.GetWindowRectangle();
IList<XYZ> corners = uiview.GetZoomCorners();
XYZ p = corners[0];
XYZ q = corners[1];

XYZ v = q - p;
XYZ center = p + 0.5 * v;
v *= 0.45;
p = center - v;
q = center + v;

uiview.ZoomAndCenterRectangle( p, q );
```

WinTooltip

- UIView
- Windows device coordinates
- ReferenceIntersector
- Custom tooltip

Summary and Further Reading

Class Summary

- Revit 2013 API add-in integration functionality
- We looked in more depth at
  - Document management and View API
  - Progress bar notifications
  - Options dialogue WPF custom extensions
  - Embedding and controlling a Revit view
  - UIView and Windows device coordinates
  - Drag and drop

Materials

- Handout
- Slide deck
- Sample Code
  - RevitUiApiNews AU sample: Options tab, drag and drop API, preview control, progress watcher, UIView
  - WinTooltip: custom tooltip, UIView, Windows coordinates, Idling, ReferenceIntersector
  - ProgressNotifier SDK sample
  - UIAPI SDK sample

Learning More

- Revit Developer Center: DevTV and My First Plugin Introduction, SDK, Samples, API Help
  http://www.autodesk.com/developrevit
- Developer Guide and Online Help
  http://www.autodesk.com/revitapi-wikihelp
- Revit API Trainings, Webcasts and Archives
  http://www.autodesk.com/apitraining > Revit API
- Discussion Group
  http://discussion.autodesk.com > Revit Architecture > Revit API
- API Training Classes
  http://www.autodesk.com/apitraining
- ADN AEC DevBlog
  http://adndevblog.typepad.com/aec
- The Building Coder, Jeremy Tammik's Revit API Blog
  http://thebuildingcoder.typepad.com
- ADN, The Autodesk Developer Network
  http://www.autodesk.com/joinadn and http://www.autodesk.com/adnopen
- DevHelp Online for ADN members
  http://adn.autodesk.com