Customizing Autodesk® Navisworks® 2013 with the .NET API

Simon Bee
Software Development Manager IPG-PDC Group
Class Summary

This class will introduce you to customizing Autodesk Navisworks 2013 using the new .NET API. It will cover how to create add-ins that sit inside the product and allow you to integrate with your own programs, how to automate the product to perform process-intensive and repetitive tasks, and how to embed the .NET controls in your own application to leverage the powers of Navisworks.
Learning Objectives

At the end of this class, you will be able to:

- Write a simple .NET plugin for Autodesk Navisworks Manage 2013.
- Understand how to automate Autodesk Navisworks.
- Write a simple application that uses the Autodesk Navisworks .NET controls.
Introduction to the Navisworks .NET API
About Navisworks
Navisworks APIs

NWcreate

.NET API

COM API
Design Philosophy

“Easy things should be easy, hard things should be possible”
Applications of the .NET API

Plugins

Automation

Control-based Applications
Assemblies

Autodesk.Navisworks.Api

Autodesk.Navisworks.Automation

Autodesk.Navisworks.Controls

Autodesk.Navisworks.ComApi
The Main Navisworks .NET API
Autodesk.Navisworks.Api

Create Plugins
Load and Save Documents
Query and Manipulate Models
Main API Classes
Main API Classes

Autodesk.Navisworks.Api.Application
Autodesk.Navisworks.Api.Model
Autodesk.Navisworks.Api.ModelItem
Querying The Model

Active Document

```
Document doc = Application.ActiveDocument;
```

Get First Model

```
Model model = doc.Models[0];
```

Get Root Item

```
ModelItem root = model.RootItem;
```

Query Item

```
bool is_hidden = root.IsHidden;
```
Object Properties
Object Properties

Autodesk.Navisworks.Api.PropertyCategory

Autodesk.Navisworks.Api.DataProperty
Access Object Properties

Finding Property Category

```csharp
PropertyCategory prop_cat = root.PropertyCategories.FindCategoryByName(
    PropertyCategoryNames.Item);
```

Accessing Property Value

```csharp
DataProperty prop = prop_cat.Properties.FindPropertyByName(
    DataPropertyNames.ItemName);
string value = prop.Value.ToString();
```
Finding Items

Search

```csharp
Search s = new Search();
ModelItemCollection searchResults = s.FindAll(false);
```

Iteration

```csharp
foreach (ModelItem item in Application.ActiveDocument.CurrentSelection.SelectedItems)
{
    // Examine item here.
}
```

LINQ

```csharp
IEnumerable<ModelItem> items =
    from item in Application.ActiveDocument.Models.GetRootItems().DescendantsAndSelf
    where ItemPrice(item) > 100
    select item;
```
Searching The Model
Using The Search Class

Create A Search

```csharp
Search search = new Search();
```

Add Selection and Conditions

```csharp
search.Selection.SelectAll();
search.SearchConditions.Add(
    SearchCondition.HasPropertyByName("LcRevitData",
        "LcRevitPropertyElementCategory"),
    EqualValue(VariantData.FromDisplayString("Stairs")));
```

Execute Search

```csharp
ModelItemCollection items = search.FindAll(Application.ActiveDocument, false);
Application.ActiveDocument.CurrentSelection.CopyFrom(items);
```
Mixing Search Methods

Combine Search With LINQ

```csharp
IEnumerable<ModelItem> expensive_items =
    from item in search.FindIncremental(Application.ActiveDocument, false);
    where ItemPrice(item) > 100
    select item;
```

Modifying The Model
Document Parts

- `Document.CurrentSelection`
- `Document.Models`
- `Document.Tool`
- `Document.SelectionSets`

- `Application.ActiveDocument.Models`
- `Application.ActiveDocument.Tool`
- `Application.ActiveDocument.SelectionSets`
Modifying The Model

Change Current Selection

```csharp
IEnumerable<ModelItem> items = Application.ActiveDocument.Models.GetRootItems().DescendantsAndSelf().Where(x => x.IsHidden = false);
Application.ActiveDocument.CurrentSelection.CopyFrom(items);
```

Override Colour

```csharp
Application.ActiveDocument.Models.OverridePermanentColor(items, Color.Red);
```

Change Tool

```csharp
Application.ActiveDocument.Tool = Tool.Select;
```
Responding To Events

DocumentPart Events

- DocumentCurrentSelection Changed
- DocumentModels

- ModelGeometryMaterialChanged

Event Handlers

```csharp
    new EventHandler<EventArgs>((CurrentSelection_Changed);

private void CurrentSelection_Changed(object sender, EventArgs e)
{
    // Respond to event...
}
```
Creating Plugins
AddInPlugin

```csharp
[PluginAttribute("MyPlugin", "ADSK", ToolTip = "My Plugin", DisplayName = "My Plugin")]
{
    public override int Execute(params string[] parameters)
    {
        //
        // Do stuff here!
        //
        return 0;
    }
}
```
Plugins In Action

Let’s Write Some Code!
The Navisworks Automation API
Autodesk.Navisworks.Automation

Automate Navisworks
Perform Batch Jobs
static void Main(string[] args)
{
    NavisworksApplication app = new NavisworksApplication();
    app.OpenFile(@"C:\file\A.nwd");
    app.AppendFile(@"C:\file\B.nwd");
    app.ExecuteAddInPlugin("Colouriser.ADSK");
    app.SaveFile(@"C:\files\Combined.nwd");
}
Automation In Action

Time To Automate Navisworks...
The Navisworks .NET Controls
Autodesk.Navisworks.Controls

.NET Controls for 3rd Party Applications

Utilise Large Model Capability
Main Control Classes

The Main Controls
Controls In Action

A Very Simple Component Viewer
The Navisworks COM Interop API
Autodesk.Navisworks.ComApi

Autodesk.Navisworks.Api

Microsoft .net

Autodesk.Navisworks.Interop.ComApi

COM
Using COM Interop

Accessing Saved Views

```csharp
InwOpState10 state = ComApiBridge.State;
InwSavedViewsColl views = state.SavedViews();
foreach (InwOpSavedView view in views)
{
    comboBox1.Items.Add(view.name);
}
```

Converting to .NET API

```csharp
InwOpSelectionSet set = (InwOpSelectionSet)sets[1];
ModelItemCollection items =
    ComApiBridge.ToModelItemCollection(set.selection);
Application.ActiveDocument.CurrentSelection.CopyFrom(items);
```
Any questions?