```
1  using AqlaSerializer;
2  using Autodesk.Revit.DB;
3  using Autodesk.Revit.UI;
4  using System;
5  using System.Collections.Generic;
6  using System.IO;
7  using System.Linq;
8  using System.Windows;
9  using System.Windows.Controls;
10 using System.Windows.Input;
11 using RvtDB = Autodesk.Revit.DB;
12
13 //https://www.devexpress.com/Support/Center/Example/Details/T192348
14 //https://www.devexpress.com/Support/Center/Example/Details/E1626
15 //https://documentation.devexpress.com/#WPF/
     DevExpressXpfNavBarNavBarGroup_ImageSourcetopic
16 //https://documentation.devexpress.com/#WPF/CustomDocument6474
17
18
19
20 namespace FacadeAssistantSQL
21 {
22     /// <summary>
23     /// Interaction logic for ProcessElements.xaml
24     /// </summary>
25     public partial class ProcessElements : UserControl
26     {
27         private UIApplication uiapp;
28         private UIDocument uidoc;
29         private Document doc;
30
31         Progress<int> progress;
32         private bool isRealTimeProgress = true;
33         private UC_Process CurrentProcess = UC_Process.Other;
34
35         private enum UC_Process : int
36         {
37             Other = 0,
38             DataGrid_Loading_Level = 1,
39             DataGrid_Loading_CurtainSystem = 2,
40             DataGrid_Loading_Wall = 3,
41             DataGrid_Loading_Panel = 4,
42             DataGrid_Loading_SubElement = 5
43         }
44
45         public ProcessElements(ExternalCommandData commandData)
46         {
47             //DevExpress.Xpf.Core.ApplicationThemeHelper.ApplicationThemeName
                = DevExpress.Xpf.Core.Theme.Office2016ColorfulName;
48             //
               DevExpress.Xpf.Core.ApplicationThemeHelper.UseLegacyDefaultThem
               e = true;
49
50             InitializeComponent();
51             InitializeCommand();
52             uiapp = commandData.Application;
```

```
53              uidoc = uiapp.ActiveUIDocument;
54              doc = uidoc.Document;
55              Global.DataFile = Path.Combine(Path.GetDirectoryName
                  (doc.PathName), $"{Path.GetFileNameWithoutExtension
                  (doc.PathName)}.data");
56
57              checkRealTimeProgress.IsChecked = isRealTimeProgress;
58
59              if(Global.DocContent is null)
60              {
61                  Global.DocContent = new DocumentContent();
62                  if (File.Exists(Global.DataFile))
63                  {
64                      using (Stream stream = new FileStream(Global.DataFile,
                        FileMode.Open, FileAccess.Read, FileShare.Read))
65                      {
66                          Global.DocContent =
                        Serializer.Deserialize<DocumentContent>(stream);
67                      }
68                      if (Global.DocContent.LevelDictionary.Count > 0)
                        Global.DocContent.IsLevelsLoaded = true;
69                      if (Global.DocContent.CurtainSystemList.Count > 0)
                        Global.DocContent.IsCurtainsystemsLoaded = true;
70                      if (Global.DocContent.WallList.Count > 0)
                        Global.DocContent.IsWallsLoaded = true;
71                      if (Global.DocContent.CurtainPanelList.Count > 0)
                        Global.DocContent.IsCurtainPanelsLoaded = true;
72                      if (Global.DocContent.ScheduleElementList.Count > 0)
                        Global.DocContent.IsDeepElementsInPanelsLoaded = true;
73                  }
74              }
75              InitProjectInfo();
76          }
77
78          #region 初始化 Command
79          private RoutedCommand cmdLoadLevel = new RoutedCommand();
80          private RoutedCommand cmdLoadCurtainSystem = new RoutedCommand();
81          private RoutedCommand cmdLoadWall = new RoutedCommand();
82          private RoutedCommand cmdLoadPanel = new RoutedCommand();
83          private RoutedCommand cmdLoadSubElement = new RoutedCommand();
84
85          private RoutedCommand cmdNavLevel = new RoutedCommand();
86          private RoutedCommand cmdNavCurtainSystem = new RoutedCommand();
87          private RoutedCommand cmdNavWall = new RoutedCommand();
88          private RoutedCommand cmdNavPanel = new RoutedCommand();
89          private RoutedCommand cmdNavSubElement = new RoutedCommand();
90
91          private RoutedCommand cmdApplyParameters = new RoutedCommand();
92
93          private void InitializeCommand()
94          {
95              CommandBinding cbLoadLevel = new CommandBinding(cmdLoadLevel,
                  cbLoadlevel_Executed, (sender, e) => { e.CanExecute = true;
                  e.Handled = true; });
96              CommandBinding cbLoadCurtainsystem = new CommandBinding
                  (cmdLoadCurtainSystem, cbLoadCurtainSystem_Executed, (sender,
```

```
                e) => { e.CanExecute = true; e.Handled = true; });
97              CommandBinding cbLoadWall = new CommandBinding(cmdLoadWall,
                  cbLoadWall_Executed, (sender, e) => { e.CanExecute = true;
                  e.Handled = true; });
98              CommandBinding cbLoadPanel = new CommandBinding(cmdLoadPanel,
                  cbLoadPanel_Executed, (sender, e) => { e.CanExecute = true;
                  e.Handled = true; });
99              CommandBinding cbLoadSubelement = new CommandBinding
                  (cmdLoadSubElement, cbLoadSubElement_Executed, (sender, e) =>
                  { e.CanExecute = true; e.Handled = true; });
100
101             CommandBinding cbNavLevel = new CommandBinding(cmdNavLevel,
                  cbNavlevel_Executed, (sender, e) => { e.CanExecute = true;
                  e.Handled = true; });
102             CommandBinding cbNavCurtainSystem = new CommandBinding
                  (cmdNavCurtainSystem, cbNavCurtainSystem_Executed, (sender, e)
                  => { e.CanExecute = true; e.Handled = true; });
103             CommandBinding cbNavwall = new CommandBinding(cmdNavWall,
                  cbNavWall_Executed, (sender, e) => { e.CanExecute = true;
                  e.Handled = true; });
104             CommandBinding cbNavPanel = new CommandBinding(cmdNavPanel,
                  cbNavPanel_Executed, (sender, e) => { e.CanExecute = true;
                  e.Handled = true; });
105             CommandBinding cbNavSubelement = new CommandBinding
                  (cmdNavSubElement, cbNavSubElement_Executed, (sender, e) =>
                  { e.CanExecute = true; e.Handled = true; });
106
107             CommandBinding cbApplyParameters = new CommandBinding
                  (cmdApplyParameters, cbApplyParameters_Executed, (sender, e) =>
                   { e.CanExecute = true; e.Handled = true; });
108
109             bnDataLevel.Command = cmdLoadLevel;
110             bnDataCurtainSystem.Command = cmdLoadCurtainSystem;
111             bnDataWall.Command = cmdLoadWall;
112             bnDataPanel.Command = cmdLoadPanel;
113             bnDataSubElement.Command = cmdLoadSubElement;
114
115             bnApplyParameters.Command = cmdApplyParameters;
116
117             ProcEle.CommandBindings.AddRange(new CommandBinding[]
                  { cbLoadLevel, cbLoadCurtainsystem, cbLoadWall, cbLoadPanel,
                  cbLoadSubelement });
118             ProcEle.CommandBindings.AddRange(new CommandBinding[]
                  { cbNavLevel, cbNavCurtainSystem, cbNavwall, cbNavPanel,
                  cbNavSubelement });
119             ProcEle.CommandBindings.Add(cbApplyParameters);
120         }
121
122         private void cbApplyParameters_Executed(object sender,
              ExecutedRoutedEventArgs e)
123         {
124             e.Handled = true;
125             ApplyParameters_CurtainPanels();
126             ApplyParameters_SubElementsInPanels();
127
128             //ContentSerialize();
```

```
129         }
130
131         private void cbNavSubElement_Executed(object sender,
              ExecutedRoutedEventArgs e)
132         {
133             #region 子構件分区分组列表
134             CurrentProcess = UC_Process.DataGrid_Loading_SubElement;
135
136             //var oldsrc = e.Parameter as IGrouping<string,
                  GeneralElementInfo>;
137             var src = (e.Parameter as IGrouping<string,
                  GeneralElementInfo>).Join(Global.DocContent.CurtainPanelList,
138                 x => x.INF_HostCurtainPanel.INF_ElementId,
139                 y => y.INF_ElementId,
140                 (x,y)=> new
141                 {
142                     Element_ID = x.INF_ElementId,
143                     Element_Name = x.INF_Name,
144                     Element_Type = x.INF_Type,
145                     Element_System = x.INF_System,
146                     Element_Direction = x.INF_Direction,
147                     Element_Level = x.INF_Level,
148                     Panel_ID = y.INF_ElementId,
149                     Panel_ZoneCode = y.INF_ZoneCode,
150                     Panel_Code = y.INF_AutoID,
151                     Element_ZoneID = x.INF_ZoneID,
152                     Element_ZoneCode = x.INF_ZoneCode,
153                     Element_Code = x.INF_AutoID
154                 });
155             gridCenter.ItemsSource = src;
156             /**
157             CurtainSystem __cs = doc.GetElement(new ElementId
                  (cs.INF_ElementId)) as CurtainSystem;
158             uidoc.Selection.Elements.Add(__cs);
159             var __bxyz = __cs.get_BoundingBox(uidoc.ActiveView);
160             uidoc.GetOpenUIViews().FirstOrDefault(v => v.ViewId.Equals
                  (doc.ActiveView.Id)).ZoomAndCenterRectangle(__bxyz.Min,
                  __bxyz.Max);
161             uidoc.RefreshActiveView();
162             **/
163             #endregion
164         }
165
166         private void cbNavPanel_Executed(object sender,
              ExecutedRoutedEventArgs e)
167         {
168             #region 嵌板分区分组列表
169             CurrentProcess = UC_Process.DataGrid_Loading_Panel;
170             gridCenter.ItemsSource = e.Parameter as IGrouping<string,
                  CurtainPanelInfo>;
171             gridCenter.Tag = "PANEL";
172             /**
173             CurtainSystem __cs = doc.GetElement(new ElementId
                  (cs.INF_ElementId)) as CurtainSystem;
174             uidoc.Selection.Elements.Add(__cs);
175             var __bxyz = __cs.get_BoundingBox(uidoc.ActiveView);
```

```
176             uidoc.GetOpenUIViews().FirstOrDefault(v => v.ViewId.Equals
                  (doc.ActiveView.Id)).ZoomAndCenterRectangle(__bxyz.Min,
                  __bxyz.Max);
177             uidoc.RefreshActiveView();
178             **/
179             #endregion
180         }
181
182         private void cbNavWall_Executed(object sender,
              ExecutedRoutedEventArgs e)
183         {
184             #region 墙分区分组列表
185             CurrentProcess = UC_Process.DataGrid_Loading_Wall;
186             gridCenter.ItemsSource = e.Parameter as IGrouping<int, WallInfo>;
187             gridCenter.Tag = "WALL";
188             /**
189             CurtainSystem __cs = doc.GetElement(new ElementId
                  (cs.INF_ElementId)) as CurtainSystem;
190             uidoc.Selection.Elements.Add(__cs);
191             var __bxyz = __cs.get_BoundingBox(uidoc.ActiveView);
192             uidoc.GetOpenUIViews().FirstOrDefault(v => v.ViewId.Equals
                  (doc.ActiveView.Id)).ZoomAndCenterRectangle(__bxyz.Min,
                  __bxyz.Max);
193             uidoc.RefreshActiveView();
194             **/
195             #endregion
196         }
197
198         private void cbNavCurtainSystem_Executed(object sender,
              ExecutedRoutedEventArgs e)
199         {
200             #region 幕墙系統分区分组列表
201             CurrentProcess = UC_Process.DataGrid_Loading_Level;
202             gridCenter.ItemsSource = e.Parameter as IGrouping<int,
                  CurtainSystemInfo>;
203             gridCenter.Tag = "CS";
204             /**
205             CurtainSystem __cs = doc.GetElement(new ElementId
                  (cs.INF_ElementId)) as CurtainSystem;
206             uidoc.Selection.Elements.Add(__cs);
207             var __bxyz = __cs.get_BoundingBox(uidoc.ActiveView);
208             uidoc.GetOpenUIViews().FirstOrDefault(v => v.ViewId.Equals
                  (doc.ActiveView.Id)).ZoomAndCenterRectangle(__bxyz.Min,
                  __bxyz.Max);
209             uidoc.RefreshActiveView();
210             **/
211             #endregion
212         }
213
214         private void cbNavlevel_Executed(object sender,
              ExecutedRoutedEventArgs e)
215         {
216             #region 標高分组列表
217             CurrentProcess = UC_Process.DataGrid_Loading_Level;
218             gridCenter.ItemsSource = Global.DocContent.LevelList;
219             gridCenter.Tag = "LEVEL";
```

```
220             #endregion
221         }
222
223         private void gridCenter_AutoGeneratingColumn(object sender,
              DataGridAutoGeneratingColumnEventArgs e)
224         {
225             switch(CurrentProcess)
226             {
227                 case UC_Process.DataGrid_Loading_Level:
228                     if (e.PropertyName == "INF_ElementId") { e.Column.Header
                        = "標高ID[ElementId]";break; }
229                     if (e.PropertyName == "INF_LevelSign") { e.Column.Header
                        = "標高編號[Sign]"; break; }
230                     if (e.PropertyName == "INF_LevelIndex") { e.Column.Header
                         = "標高樓層[Index]";break; }
231                     if (e.PropertyName == "INF_LevelElevation_Metric")
232                     {
233                         e.Column.Header = "標高數值[Elevation]";
234                         e.Column.ClipboardContentBinding.StringFormat =
                       "0.0";
235                         break;
236                     }
237                     e.Cancel = true;
238                     break;
239                 case UC_Process.DataGrid_Loading_CurtainSystem:
240                     if (e.PropertyName == "INF_ElementId") { e.Column.Header
                        = "CS-ID"; break; }
241                     if (e.PropertyName == "INF_Name") { e.Column.Header = "名
                       稱"; break; }
242                     if (e.PropertyName == "INF_CSMode") { e.Column.Header =
                       "類型"; break; }
243                     if (e.PropertyName == "INF_System") { e.Column.Header =
                       "系統"; break; }
244                     if (e.PropertyName == "INF_Direction")
245                     {
246                         e.Column.Header = "朝向";
247                         gridCenter.Columns[0].DisplayIndex = 2;
248                         break;
249                     }
250                     e.Cancel = true;
251                     break;
252                 case UC_Process.DataGrid_Loading_Wall:
253                     if (e.PropertyName == "INF_ElementId") { e.Column.Header
                        = "WALL-ID"; break; }
254                     if (e.PropertyName == "INF_Name") { e.Column.Header = "名
                       稱"; break; }
255                     if (e.PropertyName == "INF_WallMode") { e.Column.Header =
                        "類型"; break; }
256                     if (e.PropertyName == "INF_System") { e.Column.Header =
                       "系統"; break; }
257                     if (e.PropertyName == "INF_Direction")
258                     {
259                         e.Column.Header = "朝向";
260                         gridCenter.Columns[0].DisplayIndex = 2;
261                         break; }
262                     e.Cancel = true;
```

```
263                     break;
264                 case UC_Process.DataGrid_Loading_Panel:
265                     if (e.PropertyName == "INF_ElementId") { e.Column.Header = "P-ID"; break; }
266                     if (e.PropertyName == "INF_Name") { e.Column.Header = "名稱"; break; }
267                     if (e.PropertyName == "INF_Type") { e.Column.Header = "分項"; break; }
268                     if (e.PropertyName == "INF_System") { e.Column.Header = "系統"; break; }
269                     if (e.PropertyName == "INF_Direction") { e.Column.Header = "朝向"; break; }
270                     if (e.PropertyName == "INF_Level") { e.Column.Header = "樓層"; break; }
271                     if (e.PropertyName == "INF_OriginX_Metric")
                    { e.Column.Header = "原點X";
                    e.Column.ClipboardContentBinding.StringFormat = "0.0";
                    break; }
272                     if (e.PropertyName == "INF_OriginZ_Metric")
                    { e.Column.Header = "原點Z";
                    e.Column.ClipboardContentBinding.StringFormat = "0.0";
                    break; }
273                     if (e.PropertyName == "INF_Width_Metric")
                    { e.Column.Header = "寬度";
                    e.Column.ClipboardContentBinding.StringFormat = "0.0";
                    break; }
274                     if (e.PropertyName == "INF_Height_Metric")
                    { e.Column.Header = "高度";
                    e.Column.ClipboardContentBinding.StringFormat = "0.0";
                    break; }
275                     if (e.PropertyName == "INF_ZoneID") { e.Column.Header = "分區"; break; }
276                     if (e.PropertyName == "INF_ZoneCode") { e.Column.Header = "分區區號"; break; }
277                     if (e.PropertyName == "INF_AutoID")
278                     {
279                         e.Column.Header = "分區編碼";
280                         gridCenter.Columns.First(c => c.SortMemberPath == "INF_ElementId").DisplayIndex = 0;
281                         gridCenter.Columns.First(c => c.SortMemberPath == "INF_Name").DisplayIndex = 1;
282                         gridCenter.Columns.First(c => c.SortMemberPath == "INF_Type").DisplayIndex = 2;
283                         gridCenter.Columns.First(c => c.SortMemberPath == "INF_System").DisplayIndex = 3;
284                         gridCenter.Columns.First(c => c.SortMemberPath == "INF_Direction").DisplayIndex = 4;
285                         gridCenter.Columns.First(c => c.SortMemberPath == "INF_Level").DisplayIndex = 5;
286                         gridCenter.Columns.First(c => c.SortMemberPath == "INF_OriginX_Metric").DisplayIndex = 6;
287                         gridCenter.Columns.First(c => c.SortMemberPath == "INF_OriginZ_Metric").DisplayIndex = 7;
288                         gridCenter.Columns.First(c => c.SortMemberPath == "INF_Width_Metric").DisplayIndex = 8;
289                         gridCenter.Columns.First(c => c.SortMemberPath ==
```

```
                    "INF_Height_Metric").DisplayIndex = 9;
290                     gridCenter.Columns.First(c => c.SortMemberPath ==
                    "INF_ZoneID").DisplayIndex = 10;
291                     gridCenter.Columns.First(c => c.SortMemberPath ==
                    "INF_ZoneCode").DisplayIndex = 11;
292                     //Last index is automatic set.
293                     break;
294                 }
295                 e.Cancel = true;
296                 break;
297             case UC_Process.DataGrid_Loading_SubElement:
298                 if (e.PropertyName == "Element_ID") { e.Column.Header =
                    "SE-ID"; break; }
299                 if (e.PropertyName == "Element_Name") { e.Column.Header =
                     "名稱"; break; }
300                 if (e.PropertyName == "Element_Type") { e.Column.Header =
                     "分項"; break; }
301                 if (e.PropertyName == "Element_System") { e.Column.Header
                     = "系統"; break; }
302                 if (e.PropertyName == "Element_Direction")
                    { e.Column.Header = "朝向"; break; }
303                 if (e.PropertyName == "Element_Level") { e.Column.Header
                    = "樓層"; break; }
304                 if (e.PropertyName == "Panel_ID") { e.Column.Header =
                    "HOST-P-ID"; break; }
305                 if (e.PropertyName == "Panel_ZoneCode") { e.Column.Header
                     = "P-分區"; break; }
306                 if (e.PropertyName == "Panel_Code") { e.Column.Header =
                    "P-編碼"; break; }
307                 if (e.PropertyName == "Element_ZoneID") { e.Column.Header
                     = "分區號"; break; }
308                 if (e.PropertyName == "Element_ZoneCode")
                    { e.Column.Header = "分區"; break; }
309                 if (e.PropertyName == "Element_Code") { e.Column.Header =
                     "分區編碼"; break; }
310                 e.Cancel = true;
311                 break;
312             case UC_Process.Other:
313             default:
314                 break;
315         }
316     }
317
318     private void cbLoadSubElement_Executed(object sender,
          ExecutedRoutedEventArgs e)
319     {
320         e.Handled = true;
321         Load_Data_Panels_And_SubElements(true);
322     }
323
324     private void cbLoadPanel_Executed(object sender,
          ExecutedRoutedEventArgs e)
325     {
326         e.Handled = true;
327         Load_Data_Panels_And_SubElements(false);
328     }
```

```
329
330         private void cbLoadWall_Executed(object sender,
              ExecutedRoutedEventArgs e)
331         {
332             e.Handled = true;
333             Load_Data_Walls();
334             listInformation.SelectedIndex = listInformation.Items.Add($"提取当
                前模型中所有墙（幕墙）系统，共
                {Global.DocContent.WallList.Count}...");
335         }
336
337         private void cbLoadCurtainSystem_Executed(object sender,
              ExecutedRoutedEventArgs e)
338         {
339             e.Handled = true;
340             Load_Data_CurtainSystems();
341             listInformation.SelectedIndex = listInformation.Items.Add($"提取当
                前模型中所有幕墙系统，共
                {Global.DocContent.CurtainSystemList.Count}...");
342         }
343
344         private void cbLoadlevel_Executed(object sender,
              ExecutedRoutedEventArgs e)
345         {
346             e.Handled = true;
347             Load_Data_Levels();
348             listInformation.SelectedIndex = listInformation.Items.Add($"提取当
                前模型中所有标高，共 {Global.DocContent.LevelList.Count} 個...");
349         }
350
351         #endregion
352         private void InitProjectInfo()
353         {
354             if (Global.DocContent.CurtainPanelList.Count > 0)
                Global.DocContent.Lookup_CurtainPanels =
                Global.DocContent.CurtainPanelList.ToLookup(g =>
                g.INF_ZoneCode);
355             Global.DocContent.ParameterInfoList =
                ParameterHelper.RawGetProjectParametersInfo(doc);
356             SetupProjectParameters();
357             Load_Tree_Levels();
358             Load_Tree_CurtainSystems();
359             Load_Tree_Panels();
360             Load_Tree_SubElements();
361         }
362
363         private void ContentSerialize()
364         {
365             if (File.Exists(Global.DataFile)) File.Delete(Global.DataFile);
366             using (FileStream fs = new FileStream(Global.DataFile,
                FileMode.Create))
367             {
368                 Serializer.Serialize(fs, Global.DocContent);
369             }
370         }
371
```

```
372
373         private LevelInfo GetLevelByElevation(double elev)
374         {
375             if (Global.DocContent.LevelList.Count == 0) return new LevelInfo
                { INF_LevelIndex = -99 };
376             else if (elev < Global.DocContent.LevelList
                [0].INF_LevelElevation_Metric) return
                Global.DocContent.LevelList[0];
377             else return Global.DocContent.LevelList.FindLast(x =>
                x.INF_LevelElevation_Metric <= elev);
378
379         }
380
381         private void Load_Data_Levels()
382         {
383             #region 讀取標高數據
384             FilteredElementCollector collector = new FilteredElementCollector
                (doc);
385             var _levelList = collector
386                 .OfClass(typeof(Level))
387                 .ToElements()
388                 .Where(x => !((Level)x).Name.Equals("地面"))
389                 .OrderBy(elev => ((Level)elev).Elevation);
390
391             Global.DocContent.LevelList.Clear();
392             int i = 0;
393             listInformation.SelectedIndex = listInformation.Items.Add($"開始讀
                取項目標高數據...");
394             foreach (RvtDB.Element _elvl in  _levelList)
395             {
396                 Level _lvl = _elvl as Level;
397                 double _velev = Unit.CovertFromAPI
                    (DisplayUnitType.DUT_MILLIMETERS, _lvl.Elevation);
398                 var _li = new LevelInfo
399                 {
400                     INF_ElementId = _lvl.Id.IntegerValue,
401                     INF_LevelIndex = i + 1,
402                     INF_LevelElevation_Metric = _velev,
403                     INF_LevelElevation_US = _lvl.Elevation,
404                     INF_LevelSign = _lvl.Name
405                 };
406                 Global.DocContent.LevelList.Add(_li);
407                 if (isRealTimeProgress)
408                 {
409                     progressDataLevel.Text = $"{++i}/{_levelList.Count()}";
410                     listInformation.SelectedIndex = listInformation.Items.Add
                    ($"[{_li.INF_LevelSign}]:
                    {_li.INF_LevelElevation_Metric:N2}");
411                     System.Windows.Forms.Application.DoEvents();
412                 }
413             }
414             if (Global.DocContent.LevelList.Count > 0)
                Global.DocContent.IsLevelsLoaded = true;
415             listInformation.SelectedIndex = listInformation.Items.Add($"讀取項
                目標高數據 [{Global.DocContent.LevelList.Count}] 完成");
416             #endregion
```

```
417             Load_Tree_Levels();
418         }
419
420         #region 左侧树展开控制
421         private void ExpandNavTree(string expname)
422         {
423             switch(expname)
424             {
425                 case "LEVEL":
426                     expLevels.IsExpanded = true;
427                     expCurtainSystem.IsExpanded = false;
428                     expWall.IsExpanded = false;
429                     expPanel.IsExpanded = false;
430                     expSubElement.IsExpanded = false;
431                     break;
432                 case "CS":
433                     expLevels.IsExpanded = false;
434                     expCurtainSystem.IsExpanded = true;
435                     expWall.IsExpanded = false;
436                     expPanel.IsExpanded = false;
437                     expSubElement.IsExpanded = false;
438                     break;
439                 case "WALL":
440                     expLevels.IsExpanded = false;
441                     expCurtainSystem.IsExpanded = false;
442                     expWall.IsExpanded = true;
443                     expPanel.IsExpanded = false;
444                     expSubElement.IsExpanded = false;
445                     break;
446                 case "PANEL":
447                     expLevels.IsExpanded = false;
448                     expCurtainSystem.IsExpanded = false;
449                     expWall.IsExpanded = false;
450                     expPanel.IsExpanded = true;
451                     expSubElement.IsExpanded = false;
452                     break;
453                 case "SE":
454                     expLevels.IsExpanded = false;
455                     expCurtainSystem.IsExpanded = false;
456                     expWall.IsExpanded = false;
457                     expPanel.IsExpanded = false;
458                     expSubElement.IsExpanded = true;
459                     break;
460                 default:
461                     break;
462             }
463         }
464         #endregion
465
466         private void Load_Tree_Levels()
467         {
468             #region 生成標高數據分區
469             if (Global.DocContent.IsLevelsLoaded)
470                 navLevels.Items.Clear();
471             foreach (var lv in Global.DocContent.LevelList)
472             {
```

```
473                 ListBoxItem _navitem_lvl = new ListBoxItem();
474                 //_nbi_cs.ImageSource = new BitmapImage(new Uri("pack://
                      application:,,,/Images/private.png"));
475                 _navitem_lvl.Content = $"[Level {lv.INF_LevelSign}]";
476                 _navitem_lvl.Name = $"LEVEL_{lv.INF_LevelIndex:00}";
477                 _navitem_lvl.Tag = lv;
478
479                 MouseBinding mbind = new MouseBinding(cmdNavLevel, new
                      MouseGesture(MouseAction.LeftDoubleClick));
480                 mbind.CommandParameter = _navitem_lvl;
481                 mbind.CommandTarget = gridCenter;
482                 _navitem_lvl.InputBindings.Add(mbind);
483                 navLevels.Items.Add(_navitem_lvl);
484             }
485             #endregion
486             ExpandNavTree("LEVEL");
487             listInformation.SelectedIndex = listInformation.Items.Add($"生成項
                  目標高分區列表 [{Global.DocContent.LevelList.Count}]");
488         }
489
490         private void Load_Data_CurtainSystems()
491         {
492             #region 讀取幕墻系統數據
493             FilteredElementCollector collector = new FilteredElementCollector
                  (doc);
494             ElementClassFilter familyInstanceFilter = new ElementClassFilter
                  (typeof(CurtainSystem));
495             collector.WherePasses(familyInstanceFilter);
496
497             var _cscoll = collector.ToElements();
498             Global.DocContent.CurtainSystemList.Clear();
499             Global.DocContent.IsCurtainsystemsLoaded = true;
500             int i = 0;
501             foreach (RvtDB.Element _e in _cscoll)
502             {
503                 var _cs = new CurtainSystemInfo(_e as CurtainSystem);
504                 if (_cs.INF_ErrorInfo.Contains("["))
505                 {
506                     Global.DocContent.IsCurtainsystemsLoaded = false;
507                     listInformation.SelectedIndex = listInformation.Items.Add
                        ($"幕墻系統[{_cs.INF_ElementId}]錯誤：
                        {_cs.INF_ErrorInfo}...");
508                 }
509                 Global.DocContent.CurtainSystemList.Add(_cs);
510
511                 if (isRealTimeProgress)
512                 {
513                     progressDataCurtainSystem.Text = $"{++i}/{_cscoll.Count
                        ()}";
514                     System.Windows.Forms.Application.DoEvents();
515                 }
516             }
517             if (Global.DocContent.CurtainSystemList.Count <= 0)
                  Global.DocContent.IsCurtainsystemsLoaded = false;
518             listInformation.SelectedIndex = listInformation.Items.Add($"讀取幕
                  墻系統數據 [{Global.DocContent.CurtainSystemList.Count}]");
```

```
519             #endregion
520             Load_Tree_CurtainSystems();
521         }
522
523         private void Load_Tree_CurtainSystems()
524         {
525             #region 生成幕墻系統分區
526             if (Global.DocContent.IsCurtainsystemsLoaded)
527             {
528                 navCurtainSystems.Items.Clear();
529                 foreach(var groupz in
                      Global.DocContent.CurtainSystemList.OrderBy
                      (o=>o.INF_ZoneID).GroupBy(x=>x.INF_ZoneID))
530                 {
531                     ListBoxItem _navitem_cs = new ListBoxItem();
532                     _navitem_cs.Content = $"分区 [{groupz.Key:00}]";
533                     _navitem_cs.Name = $"ZONE_{groupz.Key:00}";
534                     _navitem_cs.Tag = groupz;
535
536                     MouseBinding mbind = new MouseBinding
                        (cmdNavCurtainSystem, new MouseGesture
                        (MouseAction.LeftDoubleClick));
537                     mbind.CommandParameter = groupz;
538                     mbind.CommandTarget = gridCenter;
539                     _navitem_cs.InputBindings.Add(mbind);
540                     navCurtainSystems.Items.Add(_navitem_cs);
541                 }
542             }
543             #endregion
544             ExpandNavTree("CS");
545             listInformation.SelectedIndex = listInformation.Items.Add($"生成幕
                  墻系統分區列表 [{navCurtainSystems.Items.Count}]");
546         }
547
548         private void Load_Data_Walls()
549         {
550             #region 讀取墻數據
551             FilteredElementCollector collector = new FilteredElementCollector
                  (doc);
552             ElementClassFilter familyInstanceFilter = new ElementClassFilter
                  (typeof(Wall));
553             collector.WherePasses(familyInstanceFilter);
554
555             var _wallcollection = collector.ToElements();
556             Global.DocContent.WallList.Clear();
557             Global.DocContent.IsWallsLoaded = true;
558             int i = 0;
559             foreach (RvtDB.Element _e in _wallcollection)
560             {
561                 var _wall = new WallInfo(_e as Wall);
562                 if (_wall.INF_ErrorInfo.Contains("["))
563                 {
564                     Global.DocContent.IsWallsLoaded = false;
565                     listInformation.SelectedIndex = listInformation.Items.Add
                        ($"墻（幕墻）[{_wall.INF_ElementId}]錯誤：
                        {_wall.INF_ErrorInfo}...");
```

```
566                 }
567                 Global.DocContent.WallList.Add(_wall);
568
569                 if (isRealTimeProgress)
570                 {
571                     progressDataWall.Text = $"{++i}/{_wallcollection.Count
                    ()}";
572                     System.Windows.Forms.Application.DoEvents();
573                 }
574             }
575             if (Global.DocContent.WallList.Count <= 0)
                  Global.DocContent.IsWallsLoaded = false;
576             listInformation.SelectedIndex = listInformation.Items.Add($"讀取墻
                （幕墻）數據 [{Global.DocContent.WallList.Count}]");
577             #endregion
578             Load_Tree_Wall();
579         }
580
581         private void Load_Tree_Wall()
582         {
583             #region 生成墻分區
584             if (Global.DocContent.IsWallsLoaded)
585             {
586                 navWalls.Items.Clear();
587                 foreach (var groupz in Global.DocContent.WallList.OrderBy(o
                  => o.INF_ZoneID).GroupBy(x => x.INF_ZoneID))
588                 {
589                     ListBoxItem _navitem_wall = new ListBoxItem();
590                     _navitem_wall.Content = $"分区 [{groupz.Key:00}]";
591                     _navitem_wall.Name = $"ZONE_{groupz.Key:00}";
592                     _navitem_wall.Tag = groupz;
593
594                     MouseBinding mbind = new MouseBinding(cmdNavWall, new
                    MouseGesture(MouseAction.LeftDoubleClick));
595                     mbind.CommandParameter = groupz;
596                     mbind.CommandTarget = gridCenter;
597                     _navitem_wall.InputBindings.Add(mbind);
598                     navWalls.Items.Add(_navitem_wall);
599                 }
600             }
601             #endregion
602             ExpandNavTree("WALL");
603             listInformation.SelectedIndex = listInformation.Items.Add($"生成墻
                （幕墻）分區列表 [{navWalls.Items.Count}]");
604         }
605
606         private void Load_Data_Panels_And_SubElements(bool deepcheck)
607         {
608             try
609             {
610                 #region 讀取嵌板數據
611                 FilteredElementCollector collector = new
                  FilteredElementCollector(doc);
612                 LogicalAndFilter cwpanel_InstancesFilter =
613                     new LogicalAndFilter(
614                         new ElementClassFilter(typeof(FamilyInstance)),
```

```
615                         new ElementCategoryFilter
                        (BuiltInCategory.OST_CurtainWallPanels));
616                 var eles = collector
617                     .WherePasses(cwpanel_InstancesFilter)
618                     .ToElements()
619                     .Where(x => (x as FamilyInstance).Symbol.Family.Name !=
                    "系统嵌板");
620
621                 Global.DocContent.IsCurtainPanelsLoaded = true;
622                 Global.DocContent.CurtainPanelList.Clear();
623                 Global.DocContent.ScheduleElementList.Clear();
624                 Global.DocContent.GeneralElementFullList.Clear();
625                 Global.DocContent.DeepElementList.Clear();
626                 int i = 0;
627                 foreach (var _ele in eles)
628                 {
629                     CurtainPanelInfo _gp = new CurtainPanelInfo(_ele as
                    RvtDB.Panel);
630                     //_gp.INF_Level = ElevAtLevel(_gp.INF_OriginZ_Metric).Key
                     + 1;
631                     _gp.INF_Level = GetLevelByElevation
                    (_gp.INF_OriginZ_Metric).INF_LevelIndex;
632                     //_gp.INF_ZoneCode = $"CW-{_gp.INF_Level:00}-
                    {_gp.INF_Type:00}-{_gp.INF_Direction}{_gp.INF_System}-Z
                    {_gp.INF_ZoneID:00}";
633                     _gp.INF_ZoneCode = $"CW-{_gp.INF_Level:00}-00-
                    {_gp.INF_Direction}{_gp.INF_System}-Z{_gp.INF_ZoneID:00}";
634                     Global.DocContent.CurtainPanelList.Add(_gp);
635
636                     if (isRealTimeProgress)
637                     {
638                         progressDataPanel.Text = $"[{_gp.INF_ZoneCode}], {+
                    +i}/{eles.Count()}";
639                         System.Windows.Forms.Application.DoEvents();
640                     }
641                 }
642                 if (Global.DocContent.CurtainPanelList.Count > 0)
                      Global.DocContent.IsCurtainPanelsLoaded = true;
643                 listInformation.SelectedIndex = listInformation.Items.Add
                      ($"讀取当前所有幕墙嵌板
                      [{Global.DocContent.CurtainPanelList.Count}]");
644
645                 Load_Tree_Panels();
646
647                 int __indexpanel = 0;
648                 foreach (var item in Global.DocContent.Lookup_CurtainPanels)
649                 {
650                     foreach (var p in item
651                         .OrderBy(p1 => Math.Round(p1.INF_OriginZ_Metric /
                        Constants.RVTPrecision))
652                         .ThenBy(p2 => Math.Round(p2.INF_OriginX_Metric /
                        Constants.RVTPrecision)))
653                     {
654                         __indexpanel++;
655                         p.INF_AutoID = $"CW-{p.INF_Level:00}-{p.INF_Type:00}-
                        {p.INF_Direction}{p.INF_System}-{__indexpanel:0000}";//嵌
```

```
                板编码
656                 if (deepcheck)
657                 {
658                     //查询嵌板中子构件
659                     ICollection<ElementId> p_subs = ((RvtDB.Panel)
                doc.GetElement(new ElementId
                (p.INF_ElementId))).GetSubComponentIds();
660                     p_subs.ToList<ElementId>().ForEach(id =>
661                     {
662                         ScheduleElementInfo _schedule_element = new
                ScheduleElementInfo();
663                         RvtDB.Element _element = (doc.GetElement
                (id));
664                         _schedule_element.INF_ElementId =
                id.IntegerValue;
665                         _schedule_element.INF_Name = _element.Name;
666                         _schedule_element.INF_HostCurtainPanel = p;
667
668                         #region 判断组合构件
669                         if (_schedule_element.INF_Name.Contains("挂件
                组")) _schedule_element.INF_Type = 71;
670                         else if (_schedule_element.INF_Name.Contains
                ("连接件组")) _schedule_element.INF_Type = 72;
671                         else
672                         {
673                             //非組合構件
674                             Parameter _param;
675                             if ((_param = _element.get_Parameter("分
                项")).HasValue)
676                             {
677                                 if (int.TryParse(_param.AsString(),
                out int _type)) _schedule_element.INF_Type = _type;
678                                 else
679                                 {
680                                     _schedule_element.INF_Type = -10;
681                                     _schedule_element.INF_ErrorInfo =
                 "構件[分項]參數錯誤(INF_Type)";
682                                     listInformation.SelectedIndex =
                listInformation.Items.Add
                ($"[{_schedule_element.INF_HostCurtainPanel.INF_ElementId}
                ][{_schedule_element.INF_ElementId}]:[分项]参数错误...");
683                                 }
684                             }
685                             else
686                             {
687                                 _schedule_element.INF_Type = -10;
688                                 _schedule_element.INF_ErrorInfo = "構
                件[分項]參數未設置(INF_Type)";
689                                 listInformation.SelectedIndex =
                listInformation.Items.Add
                ($"[{_schedule_element.INF_HostCurtainPanel.INF_ElementId}
                ][{_schedule_element.INF_ElementId}]:[分项]参数错误...");
690                             }
691                         }
692
693                         if (_schedule_element.INF_Type == 71 ||
```

```
                    _schedule_element.INF_Type == 72) //处理嵌套族实例
694                             {
695                                 _schedule_element.INF_HasDeepElements =
                    true;
696                                 _schedule_element.INF_DeepElements = new
                    List<DeepElementInfo>();
697                                 var _deepids = ((FamilyInstance)
                    _element).GetSubComponentIds();
698                                 foreach (ElementId sid in _deepids)
699                                 {
700                                     DeepElementInfo _deep_element = new
                    DeepElementInfo();
701                                     RvtDB.Element _de = (doc.GetElement
                    (sid));
702                                     _deep_element.INF_ElementId =
                    sid.IntegerValue;
703                                     _deep_element.INF_Name = _de.Name;
704                                     _deep_element.INF_HostCurtainPanel =
                    p;
705                                     _deep_element.INF_HostScheduleElement
                     = _schedule_element;
706
707                                     Parameter _sparam;
708                                     if ((_sparam = _de.get_Parameter("分
                    项")).HasValue)
709                                     {
710                                         if (int.TryParse(_sparam.AsString
                    (), out int _type)) _deep_element.INF_Type = _type;
711                                         else
712                                         {
713                                             _deep_element.INF_Type = -10;
714                                             listInformation.SelectedIndex
                     = listInformation.Items.Add($"[分項]參數錯誤");
715                                             listInformation.SelectedIndex
                     = listInformation.Items.Add
                    ($"[{_deep_element.INF_HostCurtainPanel.INF_ElementId}]
                    [{_deep_element.INF_ElementId}]:[分項]參数错误...");
716                                         }
717                                     }
718                                     else
719                                     {
720                                         _deep_element.INF_Type = -10;
721                                         listInformation.SelectedIndex =
                    listInformation.Items.Add($"[分項]參數未設置");
722                                         listInformation.SelectedIndex =
                    listInformation.Items.Add
                    ($"[{_deep_element.INF_HostCurtainPanel.INF_ElementId}]
                    [{_deep_element.INF_ElementId}]:[分项]参数错误...");
723                                     }
724
725                                     _deep_element.INF_Level =
                    p.INF_Level;
726                                     _deep_element.INF_System =
                    p.INF_System;
727                                     _deep_element.INF_Direction =
                    p.INF_Direction;
```

```
728                                         _deep_element.INF_ZoneID =
                    p.INF_ZoneID;
729                                         _deep_element.INF_ZoneCode =
                    p.INF_ZoneCode;
730                                         _deep_element.INF_Type_ZoneCode =
                    $"[{_deep_element.INF_ZoneCode}]
                    [{_deep_element.INF_Type:00}]";
731
    _schedule_element.INF_DeepElements.Add(_deep_element);
732                                     }
733                                 }
734                                 #endregion
735                                 _schedule_element.INF_Level = p.INF_Level;
736                                 _schedule_element.INF_System = p.INF_System;
737                                 _schedule_element.INF_Direction =
                    p.INF_Direction;
738
739                                 XYZ _xyzOrigin = ((FamilyInstance)
                    _element).GetTotalTransform().Origin;
740                                 _schedule_element.INF_OriginX_Metric =
                    Unit.CovertFromAPI(DisplayUnitType.DUT_MILLIMETERS,
                    _xyzOrigin.X);
741                                 _schedule_element.INF_OriginY_Metric =
                    Unit.CovertFromAPI(DisplayUnitType.DUT_MILLIMETERS,
                    _xyzOrigin.Y);
742                                 _schedule_element.INF_OriginZ_Metric =
                    Unit.CovertFromAPI(DisplayUnitType.DUT_MILLIMETERS,
                    _xyzOrigin.Z);
743
744                                 _schedule_element.INF_ZoneID = p.INF_ZoneID;
745                                 _schedule_element.INF_ZoneCode =
                    p.INF_ZoneCode;
746                                 _schedule_element.INF_Type_ZoneCode =
                    $"[{_schedule_element.INF_ZoneCode}]
                    [{_schedule_element.INF_Type:00}]";
747                                 Global.DocContent.ScheduleElementList.Add
                    (_schedule_element);
748                             });
749                         }
750
751                         if (isRealTimeProgress)
752                         {
753                             progressDataPanel.Text = $"{__indexpanel}/
                    {Global.DocContent.CurtainPanelList.Count}";
754                             System.Windows.Forms.Application.DoEvents();
755                         }
756                     }
757                 }
758                 listInformation.SelectedIndex = listInformation.Items.Add
                    ($"讀取当前所有嵌板構件
                    [{Global.DocContent.ScheduleElementList.Count}]");
759                 if (Global.DocContent.CurtainPanelList.Count > 0)
                    Global.DocContent.IsCurtainPanelsLoaded = true;
760                 if (Global.DocContent.GeneralElementFullList.Count > 0)
                    Global.DocContent.IsDeepElementsInPanelsLoaded = true;
761
```

```
762                 if (deepcheck)
763                 {
764                     Global.DocContent.IsDeepElementsInPanelsLoaded = true;
765 
766                     //深層构件完全清单
767                     Global.DocContent.ScheduleElementList.ForEach(s =>
768                     {
769                         Global.DocContent.GeneralElementFullList.Add(s);
770                         if (s.INF_HasDeepElements)
771                         {
772                             Global.DocContent.GeneralElementFullList.AddRange
                    (s.INF_DeepElements);
773                             Global.DocContent.DeepElementList.AddRange
                    (s.INF_DeepElements);
774                         }
775 
776                     });
777                     if (Global.DocContent.ScheduleElementList.Count > 0)
                    Global.DocContent.IsDeepElementsInPanelsLoaded = true;
778 
779                     Load_Tree_SubElements();
780                     int __indexsubelement = 0;
781                     foreach (var g in
                    Global.DocContent.Lookup_GeneralElementsFull)
782                     {
783                         foreach (var ele in g
784                             .OrderBy(e1 => Math.Round
                        (e1.INF_HostCurtainPanel.INF_OriginZ_Metric /
                        Constants.RVTPrecision))
785                             .ThenBy(e2 => Math.Round
                        (e2.INF_HostCurtainPanel.INF_OriginX_Metric /
                        Constants.RVTPrecision))
786                             .ThenBy(e3 => Math.Round(doc.GetElement(new
                        ElementId(e3.INF_ElementId)).get_BoundingBox
                        (doc.ActiveView).Min.Z / Constants.RVTPrecision))
787                             .ThenBy(e4 => Math.Round(doc.GetElement(new
                        ElementId(e4.INF_ElementId)).get_BoundingBox
                        (doc.ActiveView).Min.X / Constants.RVTPrecision)))
788                         {
789                             __indexsubelement++;
790                             ele.INF_AutoID = $"CW-{ele.INF_Level:00}-
                        {ele.INF_Type:00}-{ele.INF_Direction}{ele.INF_System}-
                        {__indexsubelement:0000}";//构件编码
791                             ////ele.INF_Element.get_Parameter("分区编码").Set
                        (ele.INF_AutoID);
792                             if (isRealTimeProgress)
793                             {
794                                 progressDataSubElement.Text =
                        $"{__indexsubelement} /
                        {Global.DocContent.GeneralElementFullList.Count}";
795                                 System.Windows.Forms.Application.DoEvents();
796                             }
797                         }
798                     }
799                     listInformation.SelectedIndex = listInformation.Items.Add
                    ($"讀取当前所有嵌板分拆構件
```

```
                    [{Global.DocContent.GeneralElementFullList.Count}]");
800                 }
801                 #endregion
802             }
803             catch (Exception ex)
804             {
805                 MessageBox.Show($"Source:{ex.Source}\nMessage:{ex.Message}
                      \nTargetSite:{ex.TargetSite}\nStackTrace:{ex.StackTrace}");
806             }
807         }
808
809         private void Load_Tree_Panels()
810         {
811             #region 生成嵌板分區
812             if (Global.DocContent.IsCurtainPanelsLoaded)
813             {
814                 navPanels.Items.Clear();
815                 Global.DocContent.Lookup_CurtainPanels =
                      Global.DocContent.CurtainPanelList.OrderBy(x =>
                      x.INF_ZoneCode).ToLookup(g => g.INF_ZoneCode);
816                 foreach (var item in Global.DocContent.Lookup_CurtainPanels)
817                 {
818                     ListBoxItem _navitem_zc = new ListBoxItem();
819                     _navitem_zc.Content = $"分區 [{item.Key}]";
820                     _navitem_zc.Name = $"ZONE_{item.Key.Replace('-', '_')}";
821                     _navitem_zc.Tag = item;
822
823                     MouseBinding mbind = new MouseBinding(cmdNavPanel, new
                        MouseGesture(MouseAction.LeftDoubleClick));
824                     mbind.CommandParameter = item;
825                     mbind.CommandTarget = gridCenter;
826                     _navitem_zc.InputBindings.Add(mbind);
827                     navPanels.Items.Add(_navitem_zc);
828                 }
829             }
830             ExpandNavTree("PANEL");
831             listInformation.SelectedIndex = listInformation.Items.Add($"生成嵌
                  板分區列表 [{navPanels.Items.Count}]");
832             #endregion
833         }
834
835
836         private void Load_Tree_SubElements()
837         {
838             #region 生成子構件分區
839             if (Global.DocContent.IsDeepElementsInPanelsLoaded)
840             {
841                 Global.DocContent.Lookup_ScheduleElements =
                      Global.DocContent.ScheduleElementList.OrderBy
                      (x=>x.INF_Type_ZoneCode).ToLookup(g =>
                      g.INF_Type_ZoneCode);
842                 Global.DocContent.Lookup_GeneralElementsFull =
                      Global.DocContent.GeneralElementFullList.OrderBy(x =>
                      x.INF_Type_ZoneCode).ToLookup(g => g.INF_Type_ZoneCode);
843                 navSubElements.Items.Clear();
844                 foreach (var g in
```

```
                    Global.DocContent.Lookup_GeneralElementsFull)
845                 {
846                     ListBoxItem _navitem_se = new ListBoxItem();
847                     _navitem_se.Content = $"分區 [{g.Key.ToString()}]";
848                     _navitem_se.Name = $"ZONE_
                        {System.Text.RegularExpressions.Regex.Replace
                        (g.Key.ToString(),@"[-|\[|\]]", "_")}";
849                     _navitem_se.Tag = g;
850
851                     MouseBinding mbind = new MouseBinding(cmdNavSubElement,
                        new MouseGesture(MouseAction.LeftDoubleClick));
852                     mbind.CommandParameter = g;
853                     mbind.CommandTarget = gridCenter;
854                     _navitem_se.InputBindings.Add(mbind);
855                     navSubElements.Items.Add(_navitem_se);
856                 }
857             }
858             ExpandNavTree("SE");
859             listInformation.SelectedIndex = listInformation.Items.Add($"生成嵌
                板構件分區列表 [{navSubElements.Items.Count}]");
860             #endregion
861         }
862
863         private void bnLoadElements_Click(object sender, RoutedEventArgs e)
864         {
865             Load_Data_Levels();
866             listInformation.SelectedIndex = listInformation.Items.Add($"提取当
                前模型中所有标高，共
                {Global.DocContent.LevelDictionary.Count}...");
867             Load_Data_CurtainSystems();
868             listInformation.SelectedIndex = listInformation.Items.Add($"提取当
                前模型中所有幕墙系统，共
                {Global.DocContent.CurtainSystemList.Count}...");
869             Load_Data_Walls();
870             listInformation.SelectedIndex = listInformation.Items.Add($"提取当
                前模型中所有墙（幕墙）系统，共
                {Global.DocContent.WallList.Count}...");
871             Load_Data_Panels_And_SubElements(false);
872             Load_Data_Panels_And_SubElements(true);
873
874             ContentSerialize();
875         }
876
877         private void bnLoadLevels_Click(object sender, RoutedEventArgs e)
878         {
879             Load_Data_Levels();
880         }
881
882         private void bnLoadCurtainSystems_Click(object sender,
              RoutedEventArgs e)
883         {
884             Load_Data_CurtainSystems();
885         }
886
887         private void bnLoadWalls_Click(object sender, RoutedEventArgs e)
888         {
```

```
889             Load_Data_Walls();
890         }
891
892         private void bnLoadPanels_Click(object sender, RoutedEventArgs e)
893         {
894             Load_Data_Panels_And_SubElements(false);
895         }
896
897         private void bnLoadSubElements_Click(object sender, RoutedEventArgs
              e)
898         {
899             Load_Data_Panels_And_SubElements(true);
900         }
901
902         private void SetupProjectParameters()
903         {
904             #region 设置项目参数
905
906             using (Transaction trans = new Transaction(doc,
                  "CreateProjectParameters"))
907             {
908                 trans.Start();
909                 #region 设置项目参数：立面朝向
910                 if (!Global.DocContent.ParameterInfoList.Exists(x => x.Name
                      == "立面朝向"))
911                 {
912                     CategorySet _catset = new CategorySet();
913                     _catset.Insert(doc.Settings.Categories.get_Item
                        (BuiltInCategory.OST_Walls));
914                     _catset.Insert(doc.Settings.Categories.get_Item
                        (BuiltInCategory.OST_CurtaSystem));
915                     _catset.Insert(doc.Settings.Categories.get_Item
                        (BuiltInCategory.OST_CurtainWallPanels));
916                     _catset.Insert(doc.Settings.Categories.get_Item
                        (BuiltInCategory.OST_GenericModel));
917                     ParameterHelper.RawCreateProjectParameter
                        (doc.Application, "立面朝向", ParameterType.Text, true,
                        _catset, BuiltInParameterGroup.INVALID, true);
918                 }
919                 #endregion
920                 #region 设置项目参数：立面系统
921                 if (!Global.DocContent.ParameterInfoList.Exists(x => x.Name
                      == "立面系统"))
922                 {
923                     CategorySet _catset = new CategorySet();
924                     _catset.Insert(doc.Settings.Categories.get_Item
                        (BuiltInCategory.OST_Walls));
925                     _catset.Insert(doc.Settings.Categories.get_Item
                        (BuiltInCategory.OST_CurtaSystem));
926                     _catset.Insert(doc.Settings.Categories.get_Item
                        (BuiltInCategory.OST_CurtainWallPanels));
927                     _catset.Insert(doc.Settings.Categories.get_Item
                        (BuiltInCategory.OST_GenericModel));
928                     ParameterHelper.RawCreateProjectParameter
                        (doc.Application, "立面系统", ParameterType.Text, true,
                        _catset, BuiltInParameterGroup.INVALID, true);
```

```
929                 }
930                 #endregion
931                 #region 设置项目参数：立面楼层
932                 if (!Global.DocContent.ParameterInfoList.Exists(x => x.Name
                      == "立面楼层"))
933                 {
934                     CategorySet _catset = new CategorySet();
935                     _catset.Insert(doc.Settings.Categories.get_Item
                      (BuiltInCategory.OST_CurtainWallPanels));
936                     _catset.Insert(doc.Settings.Categories.get_Item
                      (BuiltInCategory.OST_GenericModel));
937                     ParameterHelper.RawCreateProjectParameter
                      (doc.Application, "立面楼层", ParameterType.Integer, true,
                       _catset, BuiltInParameterGroup.INVALID, true);
938                 }
939                 #endregion
940                 #region 设置项目参数：构件分项
941                 if (!Global.DocContent.ParameterInfoList.Exists(x => x.Name
                      == "构件分项"))
942                 {
943                     CategorySet _catset = new CategorySet();
944                     _catset.Insert(doc.Settings.Categories.get_Item
                      (BuiltInCategory.OST_CurtainWallPanels));
945                     _catset.Insert(doc.Settings.Categories.get_Item
                      (BuiltInCategory.OST_GenericModel));
946                     ParameterHelper.RawCreateProjectParameter
                      (doc.Application, "构件分项", ParameterType.Integer, true,
                       _catset, BuiltInParameterGroup.INVALID, true);
947                 }
948                 #endregion
949                 #region 设置项目参数：构件子项
950                 if (!Global.DocContent.ParameterInfoList.Exists(x => x.Name
                      == "构件子项"))
951                 {
952                     CategorySet _catset = new CategorySet();
953                     _catset.Insert(doc.Settings.Categories.get_Item
                      (BuiltInCategory.OST_GenericModel));
954                     ParameterHelper.RawCreateProjectParameter
                      (doc.Application, "构件子项", ParameterType.Text, true,
                      _catset, BuiltInParameterGroup.INVALID, true);
955                 }
956                 #endregion
957                 #region 设置项目参数：加工图号
958                 if (!Global.DocContent.ParameterInfoList.Exists(x => x.Name
                      == "加工图号"))
959                 {
960                     CategorySet _catset = new CategorySet();
961                     _catset.Insert(doc.Settings.Categories.get_Item
                      (BuiltInCategory.OST_GenericModel));
962                     ParameterHelper.RawCreateProjectParameter
                      (doc.Application, "加工图号", ParameterType.Text, true,
                      _catset, BuiltInParameterGroup.INVALID, true);
963                 }
964                 #endregion
965
966                 #region 设置项目参数：分区
```

```
967                 if (!Global.DocContent.ParameterInfoList.Exists(x => x.Name
                      == "分区"))
968                 {
969                     CategorySet _catset = new CategorySet();
970                     _catset.Insert(doc.Settings.Categories.get_Item
                      (BuiltInCategory.OST_Walls));
971                     _catset.Insert(doc.Settings.Categories.get_Item
                      (BuiltInCategory.OST_CurtaSystem));
972                     _catset.Insert(doc.Settings.Categories.get_Item
                      (BuiltInCategory.OST_CurtainWallPanels));
973                     _catset.Insert(doc.Settings.Categories.get_Item
                      (BuiltInCategory.OST_GenericModel));
974                     ParameterHelper.RawCreateProjectParameter
                      (doc.Application, "分区", ParameterType.Integer, true,
                      _catset, BuiltInParameterGroup.INVALID, true);
975                 }
976                 #endregion
977
978                 #region 设置项目参数：分区区号
979                 if (!Global.DocContent.ParameterInfoList.Exists(x => x.Name
                      == "分区区号"))
980                 {
981                     CategorySet _catset = new CategorySet();
982                     _catset.Insert(doc.Settings.Categories.get_Item
                      (BuiltInCategory.OST_CurtainWallPanels));
983                     _catset.Insert(doc.Settings.Categories.get_Item
                      (BuiltInCategory.OST_GenericModel));
984                     ParameterHelper.RawCreateProjectParameter
                      (doc.Application, "分区区号", ParameterType.Text, true,
                      _catset, BuiltInParameterGroup.INVALID, true);
985                 }
986                 #endregion
987
988                 #region 设置项目参数：分区编码
989                 if (!Global.DocContent.ParameterInfoList.Exists(x => x.Name
                      == "分区编码"))
990                 {
991                     CategorySet _catset = new CategorySet();
992                     _catset.Insert(doc.Settings.Categories.get_Item
                      (BuiltInCategory.OST_CurtainWallPanels));
993                     _catset.Insert(doc.Settings.Categories.get_Item
                      (BuiltInCategory.OST_GenericModel));
994                     ParameterHelper.RawCreateProjectParameter
                      (doc.Application, "分区编码", ParameterType.Text, true,
                      _catset, BuiltInParameterGroup.INVALID, true);
995                 }
996                 #endregion
997                 #region 输出项目参数表用于设置校核
998                 //更新提取所有project parameter information
999                 Global.DocContent.ParameterInfoList =
                      ParameterHelper.RawGetProjectParametersInfo(doc);
1000                /**
1001                using (StreamWriter sw = new StreamWriter(Path.Combine
                      (Path.GetDirectoryName(doc.PathName),
                      $"ProjectParametersInfo-{DateTime.Now.ToString
                      ("yyyyMMddHHmmss")}.csv"), false, Encoding.UTF8))
```

```
1002                {
1003                    string title = string.Empty;
1004                    string rows =
                        ParameterHelper.RawParametersInfoToCSVString(paramsInfo,
                        ref title);
1005                    sw.WriteLine(title);
1006                    sw.Write(rows);
1007                }
1008                **/
1009                #endregion
1010
1011                trans.Commit();
1012
1013            }
1014            using (Transaction trans = new Transaction(doc,
                  "InitCSWALLProjectParameters"))
1015            {
1016                FilteredElementCollector collectorcs = new
                    FilteredElementCollector(doc);
1017                ElementClassFilter csFilter = new ElementClassFilter(typeof
                    (CurtainSystem));
1018                collectorcs.WherePasses(csFilter);
1019                var _cscoll = collectorcs.ToElements();
1020                FilteredElementCollector collectorwall = new
                    FilteredElementCollector(doc);
1021                ElementClassFilter wallFilter = new ElementClassFilter(typeof
                    (Wall));
1022                collectorwall.WherePasses(wallFilter);
1023                var _wallcoll = collectorwall.ToElements();
1024
1025                trans.Start();
1026                Parameter _param;
1027                foreach (var ele in _cscoll)
1028                {
1029                    if ((_param = ele.get_Parameter("系统")).HasValue)
                        ele.get_Parameter("立面系统").Set(_param.AsString());
1030                    if ((_param = ele.get_Parameter("方位")).HasValue)
                        ele.get_Parameter("立面朝向").Set(_param.AsString());
1031                    if (!(_param = ele.get_Parameter("分区")).HasValue)
                        ele.get_Parameter("分区").Set(1);
1032                }
1033                foreach (var ele in _wallcoll)
1034                {
1035                    if ((_param = ele.get_Parameter("系统")).HasValue)
                        ele.get_Parameter("立面系统").Set(_param.AsString());
1036                    if ((_param = ele.get_Parameter("方位")).HasValue)
                        ele.get_Parameter("立面朝向").Set(_param.AsString());
1037                    if (!(_param = ele.get_Parameter("分区")).HasValue)
                        ele.get_Parameter("分区").Set(1);
1038                }
1039
1040                trans.Commit();
1041
1042            }
1043            #endregion
1044        }
```

```
1045
1046        private void ApplyParameters_CurtainPanels()
1047        {
1048            //try
1049            {
1050                using (Transaction trans = new Transaction(doc,
                      "Apply_Parameters_CurtainPanels"))
1051                {
1052                    trans.Start();
1053                    int i = 0;
1054                    Global.DocContent.CurtainPanelList.ForEach(p =>
1055                    {
1056                        RvtDB.Element _element = doc.GetElement(new ElementId
                      (p.INF_ElementId));
1057                        _element.get_Parameter("立面朝向").Set
                      (p.INF_Direction);
1058                        _element.get_Parameter("立面系统").Set(p.INF_System);
1059                        _element.get_Parameter("立面楼层").Set(p.INF_Level);
1060                        _element.get_Parameter("构件分项").Set(p.INF_Type);
1061                        _element.get_Parameter("分区").Set(p.INF_ZoneID);
1062                        _element.get_Parameter("分区区号").Set
                      (p.INF_ZoneCode);
1063
1064                        if (isRealTimeProgress)
1065                        {
1066                            progressApplyPanelParameters.Text = $"{i++}/
                      {Global.DocContent.CurtainPanelList.Count}, {i * 1.0 /
                      Global.DocContent.CurtainPanelList.Count:P0}";
1067                            System.Windows.Forms.Application.DoEvents();
1068                        }
1069                    });
1070                    trans.Commit();
1071                }
1072                listInformation.SelectedIndex = listInformation.Items.Add
                    ($"寫入嵌板 [{Global.DocContent.CurtainPanelList.Count}] 參
                    數 [{Global.DocContent.CurtainPanelList.Count * 6}]");
1073            }
1074            //catch (Exception ex)
1075            {
1076                //MessageBox.Show($"Source:{ex.Source}\nMessage:{ex.Message}
                    \nTargetSite:{ex.TargetSite}\nStackTrace:{ex.StackTrace}");
1077            }
1078            //listboxOutput.SelectedIndex = listboxOutput.Items.Add($"写入[幕
                  墙嵌板]:{Global.DocContent.CurtainPanelList.Count}, 参数:
                  {Global.DocContent.CurtainPanelList.Count * 6}...");
1079        }
1080
1081        private void ApplyParameters_SubElementsInPanels()
1082        {
1083            using (Transaction trans = new Transaction(doc,
                  "Apply_Parameters_SubElementsInPanels"))
1084            {
1085                trans.Start();
1086                int i = 0;
1087                foreach (var lks in
                      Global.DocContent.Lookup_GeneralElementsFull)
```

```
1088                    foreach (var s in lks)
1089                    {
1090                        RvtDB.Element _selement = doc.GetElement(new
                        ElementId(s.INF_ElementId));
1091                        _selement.get_Parameter("立面朝向").Set
                        (s.INF_Direction);
1092                        _selement.get_Parameter("立面系统").Set(s.INF_System);
1093                        _selement.get_Parameter("立面楼层").Set(s.INF_Level);
1094                        _selement.get_Parameter("构件分项").Set(s.INF_Type);
1095                        //p.INF_Element.get_Parameter("构件子项").Set
                        (p.INF_Type);
1096                        _selement.get_Parameter("分区").Set(s.INF_ZoneID);
1097                        _selement.get_Parameter("分区区号").Set
                        (s.INF_ZoneCode);
1098                        _selement.get_Parameter("分区编码").Set(s.INF_AutoID);
1099
1100                        if (isRealTimeProgress)
1101                        {
1102                            progressApplySubElementParameters.Text = $"{i++}/
                        {Global.DocContent.GeneralElementFullList.Count}, {i *
                        1.0 / Global.DocContent.GeneralElementFullList.Count:P0}";
1103                            System.Windows.Forms.Application.DoEvents();
1104                        }
1105                    }
1106                trans.Commit();
1107            }
1108            listInformation.SelectedIndex = listInformation.Items.Add($"写入嵌
                板构件 [{Global.DocContent.ScheduleElementList.Count}] 参数
                [{Global.DocContent.ScheduleElementList.Count * 7}]");
1109        }
1110
1111        private void checkRealTimeProgress_Checked(object sender,
              RoutedEventArgs e)
1112        {
1113            isRealTimeProgress = true;
1114        }
1115
1116        private void checkRealTimeProgress_Unchecked(object sender,
              RoutedEventArgs e)
1117        {
1118            isRealTimeProgress = false;
1119        }
1120    }
1121 }
1122
```