Revit开发DevTV
—— 对象高效过滤

叶雄进
Developer Technical Services
Aug 2011. Beijing

Autodesk

关于培训者

叶雄进 Joe Ye
Joe.Ye@autodesk.com

5年工程软件开发
4年软件开发咨询

ADN AEC WorkGroup
全球 WW ADN Developer
支持APIs
- Revit
- AutoCAD Architecture
- AutoCAD

FilteredElementCollector

- 所有的对象遍历都必须要使用这个类
- FilteredElementCollector 用来过滤模型文档中的对象
- 至少有一个条件
- 可被赋予多个条件
- 添加过滤条件后，满足条件的对象集合立即就可从这个类访问

过滤操作的步骤

- 新建FilteredElementCollector 实例

  如： `var wallTypeCollector1 = new FilteredElementCollector(_doc);`

- 添加过滤条件

  如：
  ```
  wallTypeCollector1.WherePasses(new
    ElementClassFilter(typeof(WallType)));
  ```

- 访问满足条件的对象

  如：`IList<Element> wallTypes1 = wallTypeCollector1.ToElements();`

对象过滤与遍历

需要根据目标对象的特征，过滤+遍历，获得目标对象

```
var wallTypeCollector1 = new FilteredElementCollector(_doc);
 wallTypeCollector1.WherePasses(new ElementClassFilter(typeof(WallType)));
 IList<Element> wallTypes1 = wallTypeCollector1.ToElements();
```

过滤范围和过滤条件添加方法

1. 利用FilteredElementCollector 的构造函数初步一个大范围

对一个文档中进行过滤

FilteredElementCollector(Document)

只对给定视图中可见的对象过滤

FilteredElementCollector(Document, ElementId)

对一个对象集合进行过滤

FilteredElementCollector(Document, ICollection<(Of <(ElementId>)>))

2. 利用FilteredElementCollector的函数添加快捷过滤条件

后面详述

3. 通过FilteredElementCollector的WherePasses（） 函数添加任意过滤条件

添加过滤条件的快捷方法

- FilteredElementCollector的快捷方法添加过滤条件
- 快捷方法可以并联，连续添加多个条件
  - OfClass(Type)
  - OfCategory(BuiltInCategory)
  - OfCatoryId(ElementId)
  - OwnedByView(
  - ContainedInDesignOption(ElementId designOptionId)
  - WhereElementIsCurveDriven
  - WhereElementIsElementType

```
// Use OfClass method to apply an ElementClassFilter and find Levels in the document
FilteredElementCollector collector = new FilteredElementCollector(document);
ICollection<Element> levels = collector.OfClass(typeof(Level)).ToElements();
```

```
ElementType wallType5 = new FilteredElementCollector(_doc)
  .WhereElementIsElementType() // superfluous, because WallType :
  .OfClass(typeof(WallType))
  .OfCategory(BuiltInCategory.OST_Walls) // superfluous, because
  .FirstElement() as ElementType;
```

用WherePasses添加过滤条件

- FilteredElementCollector.WherePasses（ElementFilter）添加更多种过滤条件
  - 逻辑组合过滤条件
  - 任何过滤条件

```
// Find all door instances in the project by finding all elements that both belong to the door
// category and are family instances.
ElementClassFilter familyInstanceFilter = new ElementClassFilter(typeof(FamilyInstance));

// Create a category filter for Doors
ElementCategoryFilter doorsCategoryfilter = new ElementCategoryFilter(BuiltInCategory.OST_Doors);

// Create a logic And filter for all Door FamilyInstances
LogicalAndFilter doorInstancesFilter = new LogicalAndFilter(familyInstanceFilter, doorsCategoryfilter);

// Apply the filter to the elements in the active document
FilteredElementCollector collector = new FilteredElementCollector(document);
IList<Element> doors = collector.WherePasses(doorInstancesFilter).ToElements();
```

返回相同的对象集合，添加过滤条件有多种语句表达

- 多个过滤方法得到同一个对象几何
- 过滤出所有墙类型 的三种方法

```
<C#>
      var wallTypeCollector1 = new FilteredElementCollector(_doc);
      wallTypeCollector1.WherePasses(new ElementClassFilter(typeof(WallType)));
      IList<Element> wallTypes1 = wallTypeCollector1.ToElements();
</C#>
```

```
<C#>
      FilteredElementCollector wallTypeCollector2 = new FilteredElementCollector(_doc);
      wallTypeCollector2.OfClass(typeof(WallType));
</C#>
```

```
<C#>
        FilteredElementCollector wallTypeCollector3 = new
FilteredElementCollector(_doc).OfClass(typeof(WallType));
</C#>
```

如何访问过滤返回的结果（一）

- 只访问结果中的第一个对象
  - FilteredElementCollector.FirstElement（）：返回第一个对象
  - FilteredElementCollector.FirstElementId（）：返回第一个对象的Id
- 获得遍历器，以此来遍历每一个对象
  - GetElementIdIterator（）
  - GetElementIterator（）
- 直接获得结果的对象集合，转换成一个ICollection.可以直接对这个集合遍历
  - ToElementIds（）
  - ToElements（）

```
public FamilyInstance GetAColumn(Autodesk.Revit.DB.Document document)
{
    // Create a filter to find all columns
    StructuralInstanceUsageFilter columnFilter =
        new StructuralIn// Use a RoomFilter to find all room elements in the document.
                        RoomFilter filter = new RoomFilter();
    // Apply the filter
    FilteredElementColle// Apply the filter to the elements in the active document
                        FilteredElementCollector collector = new FilteredElementCollector(document);
    collector.WherePasse                        collector.WherePasses(filter);

    // Get the first col// Get results as ElementId iterator
    // Find a column wit                        FilteredElementIterator roomIdItr = collector.GetElementIdIterator();
    FamilyInstance colum                        roomIdItr.Reset();
                        while (roomIdItr.MoveNext())
    return column;      {
}                           ElementId roomId = roomIdItr.Current;
                            // Warn rooms smaller than 50 SF
                            Room room = document.get_Element(roomId) as Room;
                            if (room.Area < 50.0)
                            {
                                String prompt = "Room is too small: id = " + roomId.ToString();
                                TaskDialog.Show("Revit", prompt);
                                break;
                            }
                        }
```

如何访问过滤返回的结果（二）

- 这个类实现了 IEnumerable 接口，可直接用ForEach来遍历满足过滤条件的对象集合

```
FilteredElementCollector collector = new FilteredElementCollector(document);
collector.WherePasses(filter);
sel = app.ActiveUIDocument.Selection;
sel.Elements.Clear();
foreach (Element elem in collector)
  sel.Elements.Add(elem);
```

使用.NET 的LINQ进一步过滤

- .NET framework **L**anguage-**IN**tegrated **Q**uery, set, and transform operations.
- 使用 LINQ 来对一个集合进一步查询，针对对象属性值再过滤

```
public Element FindFamilyType_Wall_v1(string wallFamilyName, string wallTypeName)
    {
      // Narrow down a collector with class.
      var wallTypeCollector1 = new FilteredElementCollector( doc);
      wallTypeCollector1.OfClass(typeof(WallType));

      // LINQ query
      var wallTypeElems1 =
          from element in wallTypeCollector1
          where element.Name.Equals(wallTypeName)
          select element;
      Element wallType1 = null; // Result will go here.
      if (wallTypeElems1.Count<Element>() > 0)
      {
        wallType1 = wallTypeElems1.First<Element>();
      }
      return wallType1;
  }
```

过滤条件类型和功能

快速过滤条件

例如:

ElementClassFilter

ElementCategoryFilter

慢速过滤条件

例如:

FamilyInstanceFilter

AreaFilter

逻辑过滤条件，组合多个过滤条件

LogicalAndFilter

LogicalOrFilter

这些过滤条件都可以用FilteredElementCollector.WherePasses() 添加

快速过滤

过滤器名	返回结果	快捷方法
ElementCategoryFilter	获得指定category的对象	OfCategory/OfCategoryId
ElementClassFilter	获得指定类名的对象	OfClass
ElementIsElementTypeFilter	获得所有类型对象	WhereElementIsElementType WhereElementIsNotElementType
ElementOwnerViewFilter	获得只隶属指定视图中的对象	OwnedByView WhereElementIsViewIndependent
ElementDesignOptionFilter	获得指定设计选项的对象	ContainedInDesignOption
ElementIsCurveDrivenFilter	获得曲线驱动的对象（Location=LocationCurve）	WhereElementIsCurveDriven
ElementStructuralTypeFilter	获得指定结构类型的对象（梁，柱，斜梁，条基等）	none
FamilySymbolFilter	获得指定族的类型	none
ExclusionFilter	不返回指定集合中的对象	Excluding
BoundingBoxIntersectsFilter	返回对象，其bounding box与给定的区域相交）	None
BoundingBoxIsInsideFilter	返回对象（其bounding box包含在给定的区域中）	None
BoundingBoxContainsPointFilter	返回对象，其bounding box包含给定的点	None

慢速滤条件

过滤器名	输出结果	对应的快捷方法
FamilyInstanceFilter	返回指定族类型的族实例	none
ElementLevelFilter	返回位于指定楼层中的对象	none
ElementParameterFilter	返回满足参数值条件的对象	none
PrimaryDesignOptionMemberFilter	返回在主要设计选项中的对象	none
StructuralInstanceUsageFilter	返回指定结构类型的对象(墙，大梁，次梁,斜梁，…）	none
StructuralWallUsageFilter	返回指定受力类别的墙（承重，非承重，剪力墙，组合）	none
StructuralMaterialTypeFilter	返回指定的结构材料的对象（钢，混凝土，木，铝，预制混凝土）	none
RoomFilter	返回rooms	none
SpaceFilter	返回spaces	none
AreaFilter	返回 areas	none
RoomTagFilter	返回 room tags	none
SpaceTagFilter	返回 space tags	none
AreaTagFilter	返回 area tags	none
CurveElementFilter	返回指定曲线类型的对象 (model curves, symbolic curves, detail curves, etc)	none

逻辑过滤条件的使用

- 可以串联两个或两个以上过滤条件
  - LogicalAndFilter
    - LogicalAndFilter(IList(Of (ElementFilter)))
    - LogicalAndFilter(ElementFilter, ElementFilter)
  - LogicalOrFilter
    - LogicalOrFilter(IList(Of (ElementFilter)))
    - LogicalOrFilter(ElementFilter, ElementFilter)

```
// Find all door instances in the project by finding all elements that both belong to the door
// category and are family instances.
ElementClassFilter familyInstanceFilter = new ElementClassFilter(typeof(FamilyInstance));

// Create a category filter for Doors
ElementCategoryFilter doorsCategoryfilter = new ElementCategoryFilter(BuiltInCategory.OST_Doors);

// Create a logic And filter for all Door FamilyInstances
LogicalAndFilter doorInstancesFilter = new LogicalAndFilter(familyInstanceFilter, doorsCategoryfilter);

// Apply the filter to the elements in the active document
FilteredElementCollector collector = new FilteredElementCollector(document);
IList<Element> doors = collector.WherePasses(doorInstancesFilter).ToElements();
```

获取某一个楼层中的对象（快速过滤条件）

```
  [TransactionAttribute(Autodesk.Revit.Attributes.TransactionMode.Manual)]
public class RevitCommand : IExternalCommand
{
    public Result Execute(ExternalCommandData commandData, ref string messages, ElementSet elements)
    {

      UIApplication app = commandData.Application;
      Document document = app.ActiveUIDocument.Document;

      // Use ElementLevel filter to find elements by their associated level in the document
      // Find the level whose Name is "Level 1"
      FilteredElementCollector collector = new FilteredElementCollector(document);
      ICollection<Element> levels = collector.OfClass(typeof(Level)).ToElements();
      var query = from element in collector where element.Name == "Level 1" select element;// Linq query

      // Get the level id which will be used to match elements
      List<Element> level1 = query.ToList<Element>();
      ElementId levelId = level1[0].Id;

      // Find all walls on level one
      ElementLevelFilter level1Filter = new ElementLevelFilter(levelId);
      collector = new FilteredElementCollector(document);
      ICollection<Element> allWallsOnLevel1 = collector.OfClass(typeof(Wall)).WherePasses(level1Filter).ToElements();

      TaskDialog.Show("Number of rooms in Level 1", allWallsOnLevel1.Count.ToString());

      // Find all rooms not on level one: use an inverted ElementLevelFilter to match all elements not on the target level
      ElementLevelFilter notOnLevel1Filter = new ElementLevelFilter(levelId, true); // Inverted filter
      collector = new FilteredElementCollector(document);
      IList<Element> allRoomsNotOnLevel1 = collector.WherePasses(new RoomFilter()).WherePasses(notOnLevel1Filter).ToElements();


      Selection sel = app.ActiveUIDocument.Selection;
      sel.Elements.Clear();
      foreach (Element elem in allRoomsNotOnLevel1)
        sel.Elements.Add(elem);

      TaskDialog.Show("Number of rooms not in Level 1", allRoomsNotOnLevel1.Count.ToString());
      return Result.Succeeded ;
    }
}
```

参数过滤ElementParameterFilter

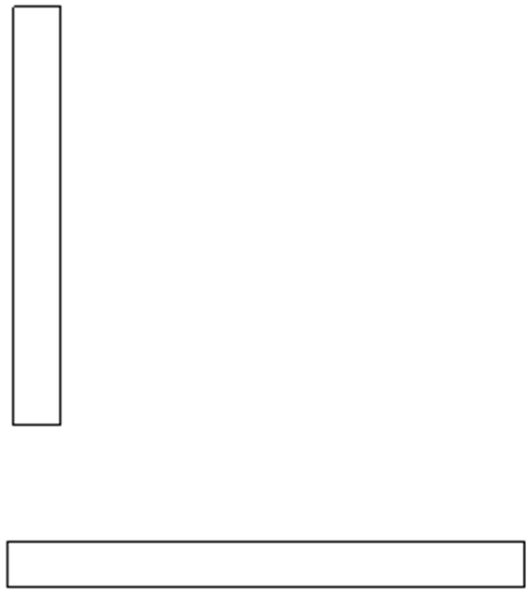
- 找到一个房间内的所有对象

```
[TransactionAttribute(Autodesk.Revit.Attributes.TransactionMode.Manual)]
public class FindFurnitureInARoom : IExternalCommand
{
    public Result Execute(ExternalCommandData commandData, ref string messages, ElementSet elements)
    {

      UIApplication app = commandData.Application;
      Document document = app.ActiveUIDocument.Document;
      //pick a room
      Selection sel = app.ActiveUIDocument.Selection ;
      Reference ref1 = sel.PickObject( ObjectType.Element, "Please pick a room");
      Room room = document.GetElement(ref1) as Room;

      ParameterValueProvider provider = new ParameterValueProvider(new ElementId(BuiltInParameter.ELEM_ROOM_ID));
      FilterNumericRuleEvaluator evaluator = new FilterNumericEquals();

      FilterElementIdRule rule = new FilterElementIdRule(provider, evaluator, room.Id);
      ElementParameterFilter filter = new ElementParameterFilter(rule);

      FilteredElementCollector collector = new FilteredElementCollector(document);
      collector.WherePasses(filter);

      sel = app.ActiveUIDocument.Selection;
      sel.Elements.Clear();
      foreach (Element elem in collector)
        sel.Elements.Add(elem);

      TaskDialog.Show("The number", "The number of components = " + collector.Count().ToString());
      return Result.Succeeded ;
    }
}
```

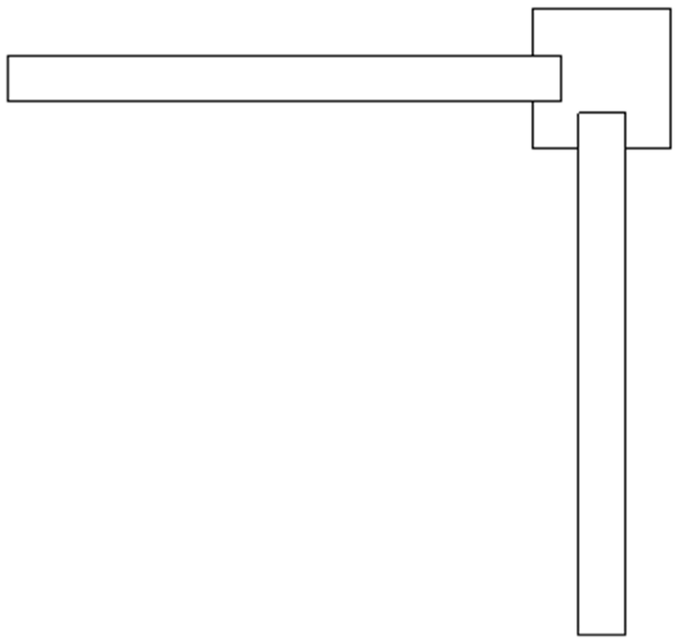
- 提示: 参数过滤条件可能比其他的类型过滤条件要快，但是这要视条件而定。毕竟这是一个慢过，使用时请按照过滤标准的复杂程度而异。

空间关系过滤条件用法示例

- 用自创几何体相交过滤法



- 用对象相交过滤法



代码

过滤与对象相交的其它对象

```
FilteredElementCollector collector = new FilteredElementCollector(doc);
ElementIntersectsElementFilter elementFilter = new ElementIntersectsElementFilter(column, false);
collector.WherePasses(elementFilter);
```

过滤与solid相交的对象

```
FilteredElementCollector collector = new FilteredElementCollector(doc);
ElementIntersectsSolidFilter solidFilter = new ElementIntersectsSolidFilter(solid);
collector.WherePasses(solidFilter);
```

对象过滤步骤

1. 首先确定你的目标对象的特征？
   - 比如：在哪个模型中，是否只在某个视图中的实体
   - 多个条件的联合
   - 常用条件的快捷方式
   - 与其他过滤对象Collector的相交或合并操作
   - 需要排除哪些对象

2. 选择用何种方式进行遍历
   - 使用遍历器Iterator
   - Element 集合 还是 ElementId 集合
   - IEnumerable

3. 获得过滤结果中的对象
   - 使用foreach来遍历每一个对象
   - LINQ 进一步限制所获得的结果集合
   - 只获得第一个对象或对象的ElementId

高效对象过滤最佳实践

- 优先使用快速过滤条件
- 慢过滤条件次之
- 使用了内置过滤技术后,考虑使用 LINQ 来更进一步限制.
- 技巧: 使用 FilteredElementCollector的快捷方法Of***（）
  - 如果你使用快捷方法，实际上是使用了快速过滤。因为慢过滤没有快捷方法.

补充

- 用空间过滤条件无法过滤轴线

Autodesk®