

## Lab3 –要素フィルタリング

Created by M. Harada, July 2010

Updated by DevTech AEC WG

Last modified: 6/9/2017

### <VB.NET>VB.NET バージョン</VB.NET>

**目的:**この実習では、Revit API のフィルタリング メカニズムを使用して、特定の要素を絞り込む方法を学習します。学習する項目は次のとおりです。

- ファミリタイプを取得する
- 特定のオブジェクトクラスのインスタンスを取得する
- 特定のファミリタイプを見つける
- 特定インスタンスを見つける

**タスク:**要素のフィルタリングの様々な手法とアプローチを理解しながらコマンドを記述していきます。フィルタリングを習熟するために、この実習を使用してください。

1. ファミリタイプをリストする(例、壁タイプ、床タイプやドアタイプ)
2. 特定のタイプのオブジェクト インスタンスをリストする(例、壁とドア)
3. 特定のファミリタイプを見つける(例、“標準壁: 一般 -200mm” 壁タイプ、“片側フラッシュ: 0915 x 2134mm” ドアタイプ)
4. 特定のインスタンスを見つける(例、“標準壁: 一般 -200mm” 壁タイプのインスタンス、片側フラッシュ: 0915 x 2134mm” ドアタイプのインスタンス、一定長より長い壁)

図 1 は、この実習で定義するコマンドで表示されるサンプル画像を示します:



図 1.様々なフィルタリングの結果を示すダイアログ

この実習の実装と確認の手順は、下記のとおりです:

1. 新しい外部コマンドを定義する
2. ファミリ タイプをリストする
3. 特定オブジェクトクラスのインスタンスをリストする
4. 特定のファミリ タイプを見つける
5. 特定のインスタンスを見つける
6. サマリ

## 1. 新しい外部コマンドを定義する

現在のプロジェクトに新しい外部コマンドを追加します。

1.1 新しいファイルを追加して、プロジェクトに新しい外部コマンドを定義します。ファイル名とクラス名は、下記のようにしてください:

- ファイル名: **3\_ElementFiltering.vb**
- コマンド クラス名: **ElementFiltering**

(繰り返しになりますが、ここで希望する名前を使用しても構いません。ただし、その場合、プロジェクト名など、このドキュメント内では記述されている名称は、自分でつけた名称で代替して参照してください)

**要求される名前空間:**

この実習で必要となる名前空間は次のとおりです:

- System.Linq
- Autodesk.Revit.DB
- Autodesk.Revit.UI
- Autodesk.Revit.ApplicationServices
- Autodesk.Revit.Attributes
- 

注意: (VB.NET のみ):VB.NET でコードを記述している場合、プロジェクト・レベルで名前空間をインポートします(つまり、プロジェクトプロパティでは、各ファイル内で明示的にインポートする必要はありません)。

1.2 Lab2 の中で行ったように、メンバ変数を定義します。例えば、DB レベルのアプリケーションとドキュメントをそれぞれ保持する m\_rvtApp と m\_rvtDoc を定義します。下記はその例です:

```
<VB.NET>
'' Element Filtering - learn about Revit element filtering
<Transaction(TransactionModeAutomatic)> _
Public Class ElementFiltering
    Implements IExternalCommand

    '' member variables
    Dim m_rvtApp As Application
```

```

Dim m_rvtDoc As Document

Public Function Execute(ByVal commandData As ExternalCommandData, _
                        ByRef message As String, _
                        ByVal elements As ElementSet) _
    As Result _
    Implements IExternalCommand.Execute

    ''' Get the access to the top most objects.
    Dim rvtUIApp As UIApplication = commandData.Application
    Dim rvtUIDoc As UIDocument = rvtUIApp.ActiveUIDocument
    m_rvtApp = rvtUIApp.Application
    m_rvtDoc = rvtUIDoc.Document

    ''' ...

    Return Result.Succeeded

End Function

End Class
</VB.NET>

```

## 2. ファミリタイプをリストする

前の実習では、要素がコンポーネント ファミリに基づくのか、システム・ファミリに基づくのかによって、クラス名とカテゴリを使用して、要素を識別するために異なるアプローチをとる必要があるということ学習しました。プロジェクト データベースに格納されたファミリ タイプのリストを取得する場合にも、同様のルールが当てはまります。

### 2.1 システム・ファミリ タイプ

Revit の要素を識別する際、それらの要素が大きな袋に入れられていて、その袋がデータベースにあると考えることができます。その中の要素にアクセスするために、クエリ(照会)をおこなう必要があります。例えば、下記のコードは、ドキュメント内のすべての WallType クラスを収集します:

```

<VB.NET>
Dim wallTypeCollector1 = New FilteredElementCollector(m_rvtDoc)
wallTypeCollector1.WherePasses( _
    New ElementClassFilter(GetType(WallType)))
Dim wallTypes1 As IList(Of Element) = wallTypeCollector1.ToElements
</VB.NET>

```

FilteredElementCollectorは、私たちが絞り込む要素を収集する「コンテナ」オブジェクトで、最初にそれを作成する必要があります。また、この場合、クラス フィルタはフィルタを通してWallTypesクラスの要素だけを収集し

ます。最終行は、フィルタされた要素コレクタを要素のリストに変換します。リストに変換することで、それ以降の要素を操作しやすくなります。

このサンプルコードでは、壁タイプの一覧を取得するためにフィルタを使用する利点を見いだせないかもしれませんが。しかし、クエリの内容がより複雑になる場合には、この方法がより便利になってくるはずです。

Revit APIは、様々なフィルタリング方式を提供しています。下記は、上記のコードの最初の2行と同じ内容ですが、OfClass() を使用しています:

```
<VB.NET>
    Dim wallTypeCollector2 = New FilteredElementCollector(m_rvtDoc)
    wallTypeCollector2.OfClass(GetType(WallType))
</VB.NET>
```

さらに、ショートカットを使用して、これを単純化することができます:

```
<VB.NET>
    Dim wallTypeCollector3 = _
        New FilteredElementCollector(m_rvtDoc).OfClass(GetType(WallType))
</VB.NET>
```

## 2.3 コンポーネント ファミリ タイプ

コンポーネント ファミリ用には、ドキュメント クラスに指定のプロパティはありません。したがって、常にフィルタリングを使用する必要があります。コンポーネント ファミリでは、要素クラスとカテゴリをチェックする必要がありました。次のコードは、ドア ファミリ タイプの一覧を得る例です。

```
<VB.NET>
    Dim doorTypeCollector = New FilteredElementCollector(m_rvtDoc)
    doorTypeCollector.OfClass(GetType(FamilySymbol))
    doorTypeCollector.OfCategory(BuiltInCategory.OST_Doors)
    Dim doorTypes As IList(Of Element) = doorTypeCollector.ToElements
</VB.NET>
```

次のコードで使用方法を確認しましょう:

```
<VB.NET>
    Public Sub ListFamilyTypes()

        ' (1) Get a list of family types available in the current rvt project.
        '
        ' For system family types, there are designated properties that
        ' allow us to directly access to the types, e.g., rvtDoc.WallTypes

        Dim s As String = String.Empty

        ' Demonstrate obsolete method use first

        'Dim wallTypes As WallTypeSet = _doc.WallTypes ' 2013
        Dim wallTypes As FilteredElementCollector = New
        FilteredElementCollector(_doc).OfClass(GetType(WallType))
```

```

For Each wType As WallType In wallTypes
    s += wType.Kind.ToString + " : " + wType.Name + vbCr
Next

TaskDialog.Show(wallTypes.Count().ToString + " Wall Types:", s)

' (1.1) Same idea applies to other system family, such as Floors, Roofs.

s = String.Empty

'Dim floorTypes As FloorTypeSet = _doc.FloorTypes
Dim floorTypes As FilteredElementCollector = New
FilteredElementCollector(_doc).OfClass(GetType(FloorType))

For Each fType As FloorType In floorTypes
    ' Family name is not in the property for floor. so use BuiltInParameter
here.
    Dim param As Parameter =
fType.Parameter(BuiltInParameter.SYMBOL_FAMILY_NAME_PARAM)
    If param IsNot Nothing Then
        s += param.AsString
    End If
    s += " : " + fType.Name + vbCr
Next

TaskDialog.Show(floorTypes.Count().ToString + " floor Types: ", s)

' (1.2a) Another approach is to use a filter. here is an example with
wall type.

Dim wallTypeCollector1 = New FilteredElementCollector(_doc)
wallTypeCollector1.WherePasses(New ElementClassFilter(GetType(WallType)))
Dim wallTypes1 As IList(Of Element) = wallTypeCollector1.ToElements

' Using a helper function to display the result here. See code below.
ShowElementList(wallTypes1, "Wall Types (by Filter): ") ' use helper
function.

' (1.2b) The following are the same as two lines above.
' These alternative forms are provided for convenience.
' Using OfClass()
'
'Dim wallTypeCollector2 = New FilteredElementCollector(_doc)
'wallTypeCollector2.OfClass(GetType(WallType))

' (1.2c) The following are the same as above for convenience
' Using short cut this time.
'
'Dim wallTypeCollector3 = New
FilteredElementCollector(_doc).OfClass(GetType(WallType))

'
' (2) Listing for component family types.
'
' For component family. it is slightly different.
' There is no designate property in the document class.

```

```

' You always need to use a filtering.
' For example, doors and windows.
' Remember for component family, you will need to check element type and
category

```

```

Dim doorTypeCollector = New FilteredElementCollector(_doc)
doorTypeCollector.OfClass(GetType(FamilySymbol))
doorTypeCollector.OfCategory(BuiltInCategory.OST_Doors)
Dim doorTypes As IList(Of Element) = doorTypeCollector.ToElements

```

```

ShowElementList(doorTypes, "Door Types (by Filter): ")

```

```

End Sub

```

```

''' <summary>
''' Helper function to display info from a list of elements passed onto.
''' </summary>

```

```

Sub ShowElementList( _
    ByVal elems As IList(Of Element), ByVal header As String)

    Dim s As String = String.Empty
    s += " - Class - Category - Name (or Family: Type Name) - Id - " + vbCr
    For Each e As Element In elems
        s += ElementToString(e)
    Next
    TaskDialog.Show(header + "(" + elems.Count.ToString() + "):", s)

```

```

End Sub

```

```

''' <summary>
''' Helper function: summarize an element information as a line of text,
''' which is composed of: class, category, name and id.
''' name will be "Family: Type" if a given element is ElementType.
''' Intended for quick viewing of list of element, for example.
''' </summary>

```

```

Function ElementToString(ByVal e As Element) As String

    If e Is Nothing Then
        Return "none"
    End If

    Dim name As String = ""

    If TypeOf e Is ElementType Then
        Dim param As Parameter =
e.Parameter(BuiltInParameter.SYMBOL_FAMILY_AND_TYPE_NAMES_PARAM)
        If param IsNot Nothing Then
            name = param.AsString
        End If
    Else
        name = e.Name
    End If

    Return e.GetType.Name + "; " + e.Category.Name + "; " _
        + name + "; " + e.Id.IntegerValue.ToString + vbCr

```

```

End Function

```

</VB.NET>

これをテストするために、Execute() メソッドから ListFamilyTypes() を呼び出すことができます。

議論:

- 壁、床およびドア以外のファミリ タイプにはどんなものがありますか？
- それらを取得するためにどのメソッドを使用することができますか？

実習:

- オブジェクトのクラスを 1 つ選んで、そのファミリタイプをすべて取得するコードを記述してください。

### 3. 特定オブジェクト クラスのインスタンスをリストする

特定のオブジェクト タイプのインスタンスの一覧を取得するためには、フィルタを使用する必要があります。ファミリタイプで学習したものと同じ考え方が、インスタンスにも当てはまります。

下記は、壁インスタンスをすべて収集する例です:

<VB.NET>

```
Dim wallCollector = _  
    New FilteredElementCollector(m_rvtDoc).OfClass(GetType(Wall))  
Dim wallList As IList(Of Element) = wallCollector.ToElements
```

</VB.NET>

次の例は、ドア インスタンスをすべて収集するものです。

<VB.NET>

```
Dim doorCollector = New FilteredElementCollector(m_rvtDoc). _  
    OfClass(GetType(FamilyInstance))  
doorCollector.OfCategory(BuiltInCategory.OST_Doors)  
Dim doorList As IList(Of Element) = doorCollector.ToElements
```

</VB.NET>

議論:

- 壁、床およびドア以外のインスタンスにはどんなものがありますか？
- それらを取得するためにどのメソッドを使用することができますか？

実習:

- 窓インスタンスをすべて取得するコードを記述してください。



## 4. 特定のファミリ タイプを見つける

このセクションでは、特定のファミリ タイプを見つける方法を学習します。次の項目を取得したいとしましょう:

- 壁タイプ — “標準壁:一般 -200mm”
- ドアタイプ — “片側フラッシュ: 0915 x 2134mm”

### 4.1 特定の名前(文字列)と一致する壁タイプを見つける

壁 タイプを取得するには、いくつかの異なる方法があります。まずは、LINQクエリを使用する方法から解説します:

<VB.NET>

```
' ' Find a specific family type for a wall with a given family and type  
' ' name. This version uses LINQ query.
```

```
Function FindFamilyType_Wall_v1( _  
    ByVal wallFamilyName As String, _  
    ByVal wallTypeName As String) As Element
```

```
    ' ' narrow down a collector with class.
```

```
    Dim wallTypeCollector1 = New FilteredElementCollector(m_rvtDoc)  
    wallTypeCollector1.OfClass(GetType(WallType))
```

```
    ' ' LINQ query
```

```
    Dim wallTypeElems1 = _  
        From element In wallTypeCollector1  
        Where element.Name.Equals(wallTypeName) _  
        Select element
```

```
    ' ' get the result.
```

```
    Dim wallType1 As Element = Nothing ' ' result will go here.
```

```
    ' ' (1) directly accessing from the query result.
```

```
    If wallTypeElems1.Count > 0 Then  
        wallType1 = wallTypeElems1.First  
    End If
```

```
    ' ' (2) if you want to get the result as a list of element,  
    ' ' here is how.
```

```
    'Dim wallTypeList1 As IList(Of Element) = wallTypeElems1.ToList()  
    'If wallTypeList1.Count > 0 Then  
    '    wallType1 = wallTypeList1(0) ' found it.  
    'End If
```

```
    Return wallType1
```

```
End Function
```

</VB.NET>

次はイテレータを使用する方法です:

<VB.NET>

```
' Find a specific family type for a wall, which is a system family.
' This version uses iteration. (cf. Developer guide 89)

Function FindFamilyType_Wall_v2( _
    ByVal wallFamilyName As String, _
    ByVal wallTypeName As String) As Element

    ' first, narrow down the collector by Class
    Dim wallTypeCollector2 = _
        New FilteredElementCollector(m_rvtDoc).OfClass(GetType(WallType))

    ' use iterator
    Dim wallTypeItr As FilteredElementIterator = _
        wallTypeCollector2.GetElementIterator
    wallTypeItr.Reset()

    Dim wallType2 As Element = Nothing

    While wallTypeItr.MoveNext
        Dim wType As WallType = wallTypeItr.Current
        ' we check two names for the match: type name and family name.
        If (wType.Name = wallTypeName) And _
            (wType.Parameter(BuiltInParameter.SYMBOL_FAMILY_NAME_PARAM). _
                AsString.Equals(wallFamilyName)) Then
            wallType2 = wType ' we found it.
            Exit While
        End If
    End While

    Return wallType2

End Function
```

</VB.NET>

#### 4.2 特定の名前(文字列)と一致する一致するドア タイプを見つける

同様に、ドア タイプについても異なる方法のアプローチが存在します。まずは、LINQ クエリを使用する方法を解説します:

<VB.NET>

```
' Find a specific family type for a door, which is a component family.
' This version uses LINQ.
'

Function FindFamilyType_Door_v1(ByVal doorFamilyName As String, ByVal
doorTypeName As String) As Element

    ' narrow down the collection with class and category.
    Dim doorFamilyCollector1 = New FilteredElementCollector(m_rvtDoc)
```

```

doorFamilyCollector1.OfClass(GetType(FamilySymbol))
doorFamilyCollector1.OfCategory(BuiltInCategory.OST_Doors)

'' parse the collection for the given name
'' using LINQ query here.
Dim doorTypeElems = _
    From element In doorFamilyCollector1
    Where element.Name.Equals(doorTypeName) And
           element.Parameter(BuiltInParameter.SYMBOL_FAMILY_NAME_PARAM).AsString.Equals(doorFamilyName) _
    Select element

'' get the result.
Dim doorType1 As Element = Nothing
'' (1) directly accessing from the query result
If doorTypeElems.Count > 0 Then '' we should have only one with the
given name. minimum error checking.
    doorType1 = doorTypeElems(0) ' found it.
End If

'' (2) if we want to get the list of element, here is how.
Dim doorTypeList As IList(Of Element) = doorTypeElems.ToList()
If doorTypeList.Count > 0 Then '' we should have only one with the
given name. minimum error checking.
    doorType1 = doorTypeList(0) ' found it.
End If

Return doorType1

End Function
</VB.NET>

```

もう一方のアプローチは、ファミリからファミリ名を参照して、次に Family.GetFamilySymbolIds プロパティからのタイプ名を調べる方法です。

```

<VB.NET>

''' <summary>
''' Find a specific family type for a door.
''' another approach will be to look up from Family, then from Family.Sym
bols property.
''' This gets more complicated although it is logical approach.
''' </summary>
Function FindFamilyType_Door_v2( _
    ByVal doorFamilyName As String, ByVal doorTypeName As String) _
    As Element

    ' (1) Find the family with the given name.

    Dim familyCollector = New FilteredElementCollector(_doc)
    familyCollector.OfClass(GetType(Family))

    ' Use the iterator

```

```

        Dim doorFamily As Family = Nothing
        Dim familyItr As FilteredElementIterator = familyCollector.GetElement
Iterator
        'familyItr.Reset()
        While (familyItr.MoveNext)
            Dim fam As Family = familyItr.Current
            ' Check name and category
            If (fam.Name = doorFamilyName) And _
                (fam.FamilyCategory.Id.IntegerValue = BuiltInCategory.OST_Doors)
Then
                ' We found the family.
                doorFamily = fam
                Exit While
            End If
        End While

        ' (2) Find the type with the given name.

        Dim doorType2 As Element = Nothing ' id of door type we are looking f
or.
        If doorFamily IsNot Nothing Then
            ' If we have a family, then proceed with finding a type under Sym
bols property.
            'Dim doorFamilySymbolSet As FamilySymbolSet = doorFamily.Symbols

            ' updated for Revit 2015
            Dim doorFamilySymbolIds As ISet(Of ElementId) = doorFamily.GetFam
ilySymbolIds()

            For Each id As ElementId In doorFamilySymbolIds
                Dim dType As FamilySymbol = TryCast(doorFamily.Document.GetEl
ement(id), FamilySymbol)
                If (dType.Name = doorTypeName) Then
                    doorType2 = dType ' Found it
                    Exit For
                End If
            Next

            End If

            Return doorType2

        End Function
</VB.NET>

```

### 4.3 より汎用的な機能を定義する

これまでは、個別の用途に向けたフィルタを定義してきました。与えられたファミリ名やタイプ名の要素を取得する機能の中で、より汎用的な関数を作成すれば、さらに便利になります。下記の関数は、引数としてドキュメント、ファミリ名、タイプ名とオプションとなるカテゴリ情報をとり、ドキュメントで見つかったファミリ タイプを返します:

<VB.NET>

```
' ' Find an element of the given type, name, and ategory(optional).

Public Shared Function FindFamilyType( _
    ByVal rvtDoc As Document, ByVal targetType As Type, _
    ByVal targetFamilyName As String, _
    ByVal targetType_name As String, _
    Optional ByVal targetCategory As BuiltInCategory = Nothing) _
    As Element

    ' ' first, narrow down to the elements of the given type and category

    Dim collector = _
        New FilteredElementCollector(rvtDoc).OfClass(targetType)
    If Not (targetCategory = Nothing) Then
        collector.OfCategory(targetCategory)
    End If

    ' ' parse the collection for the given names
    ' ' using LINQ query here.

    Dim targetElems = _
        From element In collector _
        Where element.Name.Equals(targetType_name) And _
        element.Parameter(BuiltInParameter.SYMBOL_FAMILY_NAME_PARAM). _
        AsString.Equals(targetFamilyName) _
        Select element

    ' ' put the result as a list of element fo accessibility.

    Dim elems As IList(Of Element) = targetElems.ToList()

    ' ' return the result.

    If elems.Count > 0 Then
        Return elems(0)
    End If

    Return Nothing

End Function
```

</VB.NET>

この関数を使用すると、与えられた名前を持ったファミリ タイプを以下のように見つけ出すことができます:

<VB.NET>

```
Dim wallType3 As Element = _
    FindFamilyType(m_rvtDoc, GetType(WallType), _
        "標準壁", "一般 -200mm")
Dim doorType3 As Element = _
    FindFamilyType(m_rvtDoc, GetType(FamilySymbol), _
        "片側フラッシュ", "0915 x 2134mm", BuiltInCategory.OST_Doors)
```

</VB.NET>

実習:

- 与えられた名前のファミリ タイプを取得してファミリ タイプを返す FindFamilyType() を実装してください
- FindFamilyType() を使用して、壁タイプ、ドア タイプと窓タイプから好みのものを取得してください (ファミリ名はハードコードできます)

## 5. 特定インスタンスを見つける

### 5.1 特定のファミリ タイプに該当するインスタンスを見つける

他の状況では、特定のファミリ タイプに該当するインスタンスを取得したい場面があるかもしれません。下記の関数は、クラス タイプとファミリ タイプの要素 id、オプションとなるカテゴリをとり、与えられたファミリ タイプのインスタンスのリストを返します:

<VB.NET>

```
'' Find a list of element with the given Class, family type and
'' Category (optional).
Function FindInstancesOfType( _
    ByVal targetType As Type, _
    ByVal idType As ElementId, _
    Optional ByVal targetCategory As BuiltInCategory = Nothing) _
    As IList(Of Element)

    '' narrow down to the elements of the given type and category

    Dim collector = _
        New FilteredElementCollector(m_rvtDoc).OfClass(targetType)
    If Not (targetCategory = Nothing) Then
        collector.OfCategory(targetCategory)
    End If

    '' parse the collection for the given family type id.
    '' using LINQ query here.

    Dim elems = _
        From element In collector _
```

```

        Where element.Parameter(BuiltInParameter.SYMBOL_ID_PARAM). _
            AsElementId.Equals(idType) _
        Select element

    '' put the result as a list of element fo accessibility.

    Return elems.ToList()

End Function
</VB.NET>

```

例えば、この関数を使用すると、引数で渡したファミリタイプのインスタンスのリストを、次のように見つけることができます：

```

<VB.NET>
Dim walls As IList(Of Element) = _
    FindInstancesOfType(GetType(Wall), idWallType)
Dim doors As IList(Of Element) = _
    FindInstancesOfType(GetType(FamilyInstance), idDoorType, _
        BuiltInCategory.OST_Doors)
</VB.NET>

```

## 5.2 与えられたクラスと名前で要素を見つける

一般に用いられている別のシナリオは、レベルやビューのような、Revit がサポートする要素を取得することかもしれません。次の関数は、各レベル要素などを取得するのに便利な関数になるでしょう。

```

<VB.NET>
    '' Find a list of elements with given class, name, category (optional).

    Public Shared Function FindElements( _
        ByVal rvtDoc As Document, _
        ByVal targetType As Type, _
        ByVal targetName As String, _
        Optional ByVal targetCategory As BuiltInCategory = Nothing) _
        As IList(Of Element)

        '' narrow down to the elements of the given type and category
        Dim collector = _
            New FilteredElementCollector(rvtDoc).OfClass(targetType)
        If Not (targetCategory = Nothing) Then
            collector.OfCategory(targetCategory)
        End If

        '' parse the collection for the given names
        '' using LINQ query here.

        Dim elems = _
            From element In collector _
            Where element.Name.Equals(targetName) _
            Select element
    End Function
</VB.NET>

```

```

        '' put the result as a list of element for accessibility.
        Return elems.ToList()

End Function

'' -----
'' Searches elements with given Class, Name and Category (optional),
'' and returns the first in the elements found.

Public Shared Function FindElement(ByVal rvtDoc As Document, _
    ByVal targetType As Type, _
    ByVal targetName As String, _
    Optional ByVal targetCategory As BuiltInCategory = Nothing) _
    As Element

    '' find a list of elements using the overloaded method.
    Dim elems As IList(Of Element) = _
        FindElements(rvtDoc, targetType, targetName, targetCategory)

    '' return the first one from the result.
    If elems.Count > 0 Then
        Return elems(0)
    End If

    Return Nothing

End Function
</VB.NET>

```

例えば、この関数を使用すると、引数にファミリ タイプを渡して、インスタンスのリストを、次のように見つけることができます:

```

<VB.NET>
Dim level1 As Level = FindElement(m_rvtDoc, GetType(Level), "レベル 1")
</VB.NET>

```

後の実習では、単純な家を作成するためにこの関数を使用します。

### 実習:

- 引数としてドキュメント、クラス、名前およびオプションのカテゴリをとり、与えられたクラス、名前およびカテゴリで要素のリストを返す FindElements() 関数を実装する
- FindElements() を呼び出して、リスト内の最初の要素だけを返す FindElement() を実装する
- FindElement()を使用して、与えられた名前のレベル要素を取得する(レベル名はハードコード可能)



### 5.3 パラメータ(オプション)でフィルタリングする

ここまでで、要素をフィルタリングする基礎を習得しました。具体的には、次のクラスを使用する方法を学習しました:

- FilteredElementCollector
- ElementClassFilter
- ElementCategoryFilter

さらに異なる種類のフィルタが存在します。例えば:

- BoundingBoxContainsPointFilter
- ElementDesignOptionFilter
- ElementIsCurveDrivenFilter
- ElementIsElementTypeFilter
- ElementParameterFilter
- ...

Revit API 開発者用ガイドの[「フィルタリング」](#)項目では、フィルタリングについて記述しています。詳細については、このドキュメントを参照してください。

次のコードは、パラメータ フィルタの1例です。このコードは、次のように、評価する長さと同等の壁パラメータ値をチェックするパラメータ フィルタを使用しています。

```
wall.parameter(length) > 60 フィート
```

**重要:** Revit API で使われる長さの単位はフィートです。

#### <VB.NET>

```
'' example of parameter filter.
'' find walls whose length is longer than 60 feet.
''     wall.parameter(length) > 60 feet
'' This could get more complex than looping through in terms of
'' writing a code. See page 87 of Developer guide.

Function FindLongWalls() As IList(Of Element)

    '' constant for this function.

    Const kWallLength As Double = 60.0 '' 60 feet. hard coding

    '' first, narrow down to the elements of the given type and category

    Dim collector = _
        New FilteredElementCollector(m_rvtDoc).OfClass(GetType(Wall))

    '' define a filter by parameter
    '' 1st arg - value provider

    Dim lengthParam As BuiltInParameter = _
        BuiltInParameter.CURVE_ELEM_LENGTH
    Dim iLengthParam As Integer = lengthParam
    Dim paramValueProvider = _
```

```

        New ParameterValueProvider(New ElementId(iLengthParam))

    '' 2nd - evaluator
    Dim evaluator As New FilterNumericGreater

    '' 3rd - rule value
    Dim ruleVal As Double = kWallLength

    '' 4th - epsilon
    Const eps As Double = 0.000001

    '' define a rule
    Dim filterRule = New FilterDoubleRule( _
        paramValueProvider, evaluator, ruleVal, eps)

    '' create a new filter
    Dim paramFilter = New ElementParameterFilter(filterRule)

    '' go through the filter
    Dim elems As IList(Of Element) = _
        collector.WherePasses(paramFilter).ToElements

    Return elems

End Function
</VB.NET>

```

## 6. サマリ

この実習では、要素のフィルタリング方法を学習しました。学習した項目は、次のとおりです。

- ファミリタイプをリストする
- 特定オブジェクトクラスのインスタンスをリストする
- 特定のファミリタイプを見つける
- 特定のインスタンスを見つける

次の実習では、Revit 内の要素を修正する方法を学習します。