

Lab2 – DB 要素

Created by M. Harada, July 2010

Updated by Ryuji Ogasawara

Last modified: 8/8/2025

<C#>C#バージョン</C#>

目的:この実習では、Revit 内で要素はどのように表現されているか、また、要素の情報をどのように取得するかについて学習します。学習する項目は、次のとおりです。

- 要素を識別する
- 要素の 1 セットのプロパティを取得する
- 要素の特定のプロパティを取得する
- 位置情報を取得する
- ジオメトリ情報を取得する

タスク:要素の選択をユーザに促して、要素の種類を識別し、プロパティや位置、ジオメトリ情報を表示するコマンドを記述します:

1. 要素を取得する
2. 要素とそのファミリー・タイプ(つまり)に関する基本的な情報を示すクラス名、カテゴリと要素 Id
3. 取得した要素を識別する (例えば、壁、窓あるいはドアのいずれか)
4. 要素とそのファミリー・タイプのプロパティ セットを示す
5. 要素とそのファミリー・タイプの特定のプロパティを示す
6. 要素の位置情報を示す
7. 要素(オプション)のジオメトリ情報を示す

図 1 および 2 は、この実習で定義するコマンドを実行した後に出力のサンプル画像を示します:

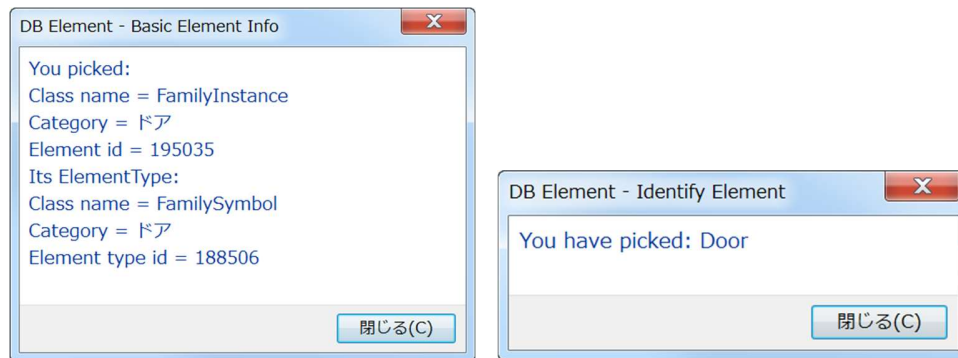


図 1.要素の基本的な情報と識別を示すダイアログ

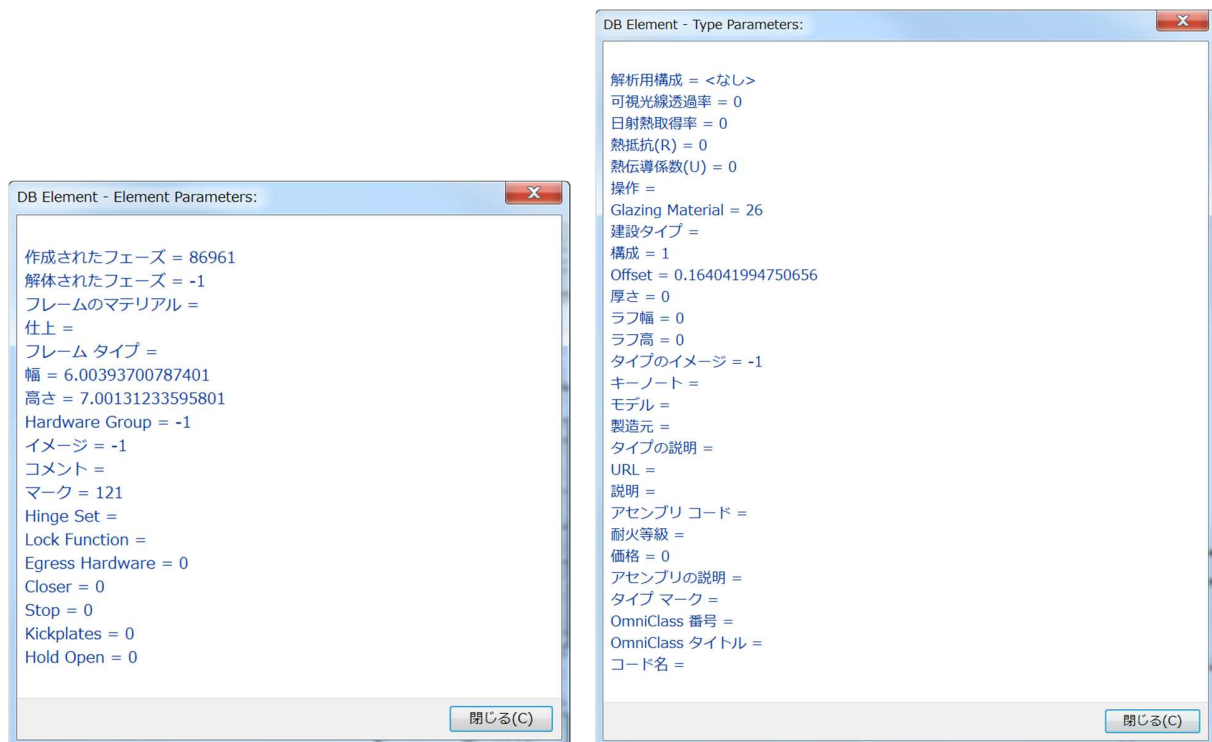


図 2.要素のプロパティ(あるいはパラメータ)とタイプを示すダイアログ

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. 要素を選択する
3. 基本的な要素の情報
4. 要素の識別
5. パラメータ
6. 位置情報
7. ジオメトリ情報(オプション)
8. 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別の外部コマンドを追加します。ここでは、新しいプロジェクトを作成せずに、既存のプロジェクトを使用してください。

1.1 新しいファイルを追加して、プロジェクトへの別の外部コマンドを定義してください。ファイル名とクラス名は次のように指定します:

- ファイル名: **2_DbElement.cs**
- コマンド クラス名: **DBElement**

(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)

必要な名前空間:

この実習で必要とする名前空間は次のとおりです:

- (System.Linq)(Lab3 で必要)
- Autodesk.Revit.DB
- Autodesk.Revit.UI

- Autodesk.Revit.ApplicationServices
- Autodesk.Revit.Attributes
- Autodesk.Revit.UI.Selection(選択処理用)

1.2 実習のトップ・レベル・オブジェクトにアクセスすることを容易にするために、ドキュメントの全体にわたってアクセス可能なトップ・レベル・オブジェクトあるいはクラスを保持するメンバ変数を定義します。Revit には DB と UI のオブジェクト間の分離の概念があります。これは Revit アプリケーションとドキュメント オブジェクトに適用されます。トップ・レベル・オブジェクトは、以下のようにアクセス可能です:

- UIApplication ← commandData.Application
- Application ← UIApplication.Application
- UIDocument ← UIApplication.ActiveUIDocument
- Document ← UIDocument.Document

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:

```
<C#>
// DB Element - learn about Revit element
[Transaction(TransactionMode.Manual)]
public class DBElement : IExternalCommand
{
    // Member variables
    Application m_rvtApp;
    Document m_rvtDoc;

    public Result Execute(ExternalCommandData commandData,
                          ref string message,
                          ElementSet elements)
    {
        // Get the access to the top most objects.
        UIApplication rvtUIApp = commandData.Application;
        UIDocument rvtUIDoc = rvtUIApp.ActiveUIDocument;
        m_rvtApp = rvtUIApp.Application;
        m_rvtDoc = rvtUIDoc.Document;

        // ...
    }
}
```

```

        return Result.Succeeded;
    }
}
</C#>

```

2. 要素を選択する

Autodesk.Revit.DB.Elementは、Revit プロジェクトデータベース内のオブジェクトの基本クラスです。ユーザインタフェース画面上の任意の要素を選択して、Revitの要素に関して学習するために、それを検査していきます。

スクリーン上でオブジェクトを選択するために、PickObject()メソッドのオーバーロードの1つを使用することができます:

- UIDocument.Selection.PickObject(ObjectType.Element、promptString)

(トレーニングの UI トピックに入る際、UI と選択のトピックを紹介します。今のところ、この実習の目的にはこれで十分です)。

次のコードは使用する際の一例です:

```

<C#>
    // (1) pick an object on a screen.
    Reference refPick = rvtUIDoc.Selection.PickObject(
        ObjectType.Element, "Pick an element");

    // we have picked something.
    Element elem = m_rvtDoc.GetElement(refPick);
</C#>

```

PickObject() は、オブジェクトの参照を返します。返されたオブジェクト参照から要素を取得することができます。

3. 基本的な要素情報

API の典型的なプログラミングや使用法では、クラス名をチェックすることで与えられたオブジェクトを識別します。このセクションでは、Revit API でも同様の方法で識別できるか確認します。

3.1 引数として要素をとる関数を記述して、与えられた要素の以下のプロパティを表示します:

- クラス名(あるいは .NET 内でのタイプ)
- カテゴリ名
- Id(要素Id)

次に、与えられた要素のファミリー・タイプで同じことをします。与えられた要素のファミリー・タイプを得るためには、最初に `Element.GetTypeId()` で `Id` を取得して、次に、`Document.GetElement(elementId)` を使用することができます。

ここでは、例えば **ShowBasicElementInfo()** 関数を使用します。下記は、上記内容を実装するサンプルコードです：

```
<C#>
public void ShowBasicElementInfo(Element elem)
{
    // let's see what kind of element we got.
    //
    string s = "You Picked:" + "\n";

    s += " Class name = " + elem.GetType().Name + "\n";
    s += " Category = " + elem.Category.Name + "\n";
    s += " Element id = " + elem.Id.ToString() + "\n" + "\n";

    // and, check its type info.
    //
    //Dim elemType As ElementType = elem.ObjectType ' this is obsolete.
    ElementId elemTypeId = elem.GetTypeId();
    ElementType elemType = (ElementType)m_rvtDoc.GetElement(elemTypeId);

    s += "Its ElementType:" + "\n";
    s += " Class name = " + elemType.GetType().Name + "\n";
    s += " Category = " + elemType.Category.Name + "\n";
    s += " Element type id = " + elemType.Id.ToString() + "\n";

    // finally show it.

    TaskDialog.Show("Basic Element Info", s);
}
</C#>
```

その後、要素を選択した直後にメインの `Execute()` メソッドからこの関数を呼び出します：

```
<C#>
    //' we have picked something.
    Element elem = refPick.Element;

    // (2) let's see what kind of element we got.
    ShowBasicElementInfo(elem);
</C#>
```

3.2 プロジェクトをビルドしてマニフェスト・ファイルを追加します。

```
<AddIn Type="Command">
  <Text>DB Element</Text>
```

```
<FullClassName>IntroCs.DBElement</FullClassName>
<Assembly>C:\...\IntroCs.dll</Assembly>
<AddInId>827AC040-6F44-4c03-82FE-292705580800</AddInId>
<VendorId>ADNP</VendorId>
<VendorDescription>Autodesk, Inc. www.autodesk.com</VendorDescription>
</AddIn>
```

“DB Element” コマンドを実行して、要素を選択してください。下記のように、選択した要素のクラス、カテゴリと id を示すダイアログが表示されるはずです(図 3)。他の要素を選択して、出力内容を観察してください。



図 3.ドアのプロパティを示すダイアログ

議論:

- 壁、ドアや窓のように、異なる要素でもクラス名とカテゴリを比較してみてください。どんな違いがありますか。
- クラス名から要素を識別できましたか？
- カテゴリから要素を識別できましたか？

周囲の受講者、同僚やインストラクタとこれらについて議論してみてください。

4. 要素の識別

前項で議論頂いたとおり、クラス名で Revit 内の要素を識別するのは十分ではありません。作図された要素によっては、下記をチェックする必要があります:

- クラス名
- カテゴリ
- 要素が要素タイプ(シンボル)であるか否か

表1 は、いくつかの要素について、要素を識別するために使われるクラス名とカテゴリの例を示しています。これらは、システム ファミリ 対 コンポーネント ファミリ、ファミリ タイプ 対 インスタンス の4つのエリアに分割されます。こ

の表から、クラス名によって識別できる要素と、またカテゴリによる識別が必要となる要素があることを理解しましょう。

- システム ファミリは、Revit に既存で組み込まれている要素です。それらには一意の定義クラスがあり、システム ファミリ要素を識別するために使用することができます。
- コンポーネント ファミリは、FamilyInstance/FamilySymbol クラスとして総括的なクラスで定義されている要素です。これらの要素はカテゴリで種類を識別できます。

	システム・ファミリ	コンポーネント ファミリ
ファミリ・タイプ	WallType FloorType	FamilySymbol & カテゴリ - ドア、窓
インスタンス	Wall Floor	FamilyInstance & カテゴリ - ドア、窓

表1. 壁、床、ドア、窓の要素を識別するクラス名およびカテゴリ

要素がRevitデータベースの中でどう表現されるか理解したところで、要素を識別するコードを追加します。下記がその例です:

```
<C#>
// identify the type of the element known to the UI.
public void IdentifyElement(Element elem)
{
    // An instance of a system family has a designated class.
    // You can use it identify the type of element.
    // e.g., walls, floors, roofs.
    //
    string s = "";

    if (elem is Wall)
    {
        s = "Wall";
    }
    else if (elem is Floor)
    {
        s = "Floor";
    }
    else if (elem is RoofBase)
    {
        s = "Roof";
    }
    else if (elem is FamilyInstance)
```



```

{
    // An instance of a component family is all FamilyInstance.
    // We'll need to further check its category.
    // e.g., Doors, Windows, Furnitures.
    if (elem.Category.Id.IntegerValue ==
        (int)BuiltInCategory.OST_Doors)
    {
        s = "Door";
    }
    else if (elem.Category.Id.IntegerValue ==
        (int)BuiltInCategory.OST_Windows)
    {
        s = "Window";
    }
    else if (elem.Category.Id.IntegerValue ==
        (int)BuiltInCategory.OST_Furniture) {
        s = "Furniture";
    }
    else
    {
        // e.g. Plant
        s = "Component family instance";
    }
}
else if (elem is HostObject)
{
    // check the base class. e.g., CeilingAndFloor.
    s = "System family instance";
}
else
{
    s = "Other";
}

s = "You have picked: " + s;

// show it.
TaskDialog.Show("Identify Element", s);
}
</C#>

```

ShowBasicElementInfo() の直後に、Execute() メソッドからこの関数を呼び出します:

```

<C#>
    // (2) let's see what kind of element we got.
    ShowBasicElementInfo(elem);

    // (3) identify each major types of element.
    IdentifyElement(elem);
</C#>

```

コードをビルドして、選択した要素を識別できる “DB Element” コマンドを再度実行してください。図4は、コマンドを実行するサンプル画像を示します。

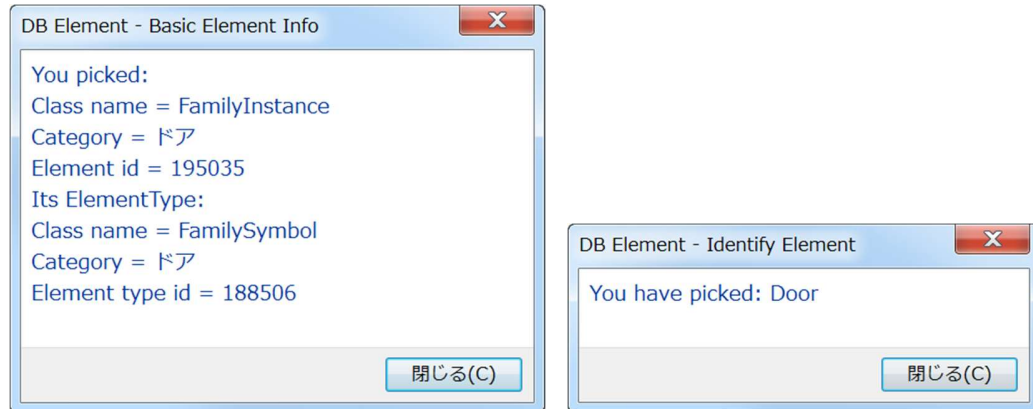


図4.要素の識別

5. パラメータ

Element クラスの Parameters プロパティは、大部分が UI 上で見る要素やファミリの「プロパティ」に相当します。Revit API には、それらのプロパティまたはパラメータにアクセスするために複数の方法があります:

- Element.get_Parameter — 一意に識別可能なパラメータID (たとえば BuiltInParameter のパラメータID) を引数に取り、戻り値としてそのパラメータを返す
- Element.LookupParameter — パラメータ名を引数に取り、単一のパラメータを返す
- Element.Paramater プロパティ — 組み込み ID、定義、GUID がわかっている場合に、一致するプロパティオブジェクトを取得できます
- Element.GetOrderedParameters — プロパティパレットで表示されるパラメータのコレクションを返す
- Element.GetParameters — パラメータ名を引数に取り、一致するパラメータのコレクションを返す

5.1 GetOrderedParameters () を介してパラメータ セットを取得する

GetOrderedParameters () を最初に見ていきます。下記のコードは、パラメータ セットを返す GetOrderedParameters () の使用方法を示しています。各パラメータにアクセスするため、単純にループすることができます。注意を払うべき主要箇所は、StorageTypeによって各パラメータの解析が必要となる点です。パラメータは、整数、倍精度実数、文字列とElementId になり得ます。StorageTypeによって、実際の値を得る方法を選ぶ必要があります。

<C#>

```
// show all the parameter values of the element
public void ShowParameters(Element elem, string header)
{

    ParameterSet paramSet = elem.GetOrderedParameters();
    string s = string.Empty;

    foreach (Parameter param in paramSet)
    {
        string name = param.Definition.Name;

        // see the helper function below
        string val = ParameterToString(param);

        s += name + " = " + val + "\n";
    }

    TaskDialog.Show(header, s);
}

// Helper function: return a string from of the given parameter.
//
public static string ParameterToString(Parameter param)
{
    string val = "none";

    if (param == null)
    {
        return val;
    }

    // to get to the parameter value, we need to pause it depending
    // on its strage type
    switch (param.StorageType)
    {
        case StorageType.Double:
            double dVal = param.AsDouble();
            val = dVal.ToString();
            break;

        case StorageType.Integer:
            int iVal = param.AsInteger();
            val = iVal.ToString();
            break;

        case StorageType.String:
            string sVal = param.AsString();
            val = sVal;
            break;

        case StorageType.ElementId:
```

```

        ElementId idVal = param.AsElementId();
        val = idVal.IntegerValue.ToString();
        break;

        case StorageType.None:
            break;

        default:
            break;
    }

    return val;
}
</C#>

```

IdentifyElement()の後で、メインの Execute() メソッドからこの関数を呼び出します。さらに、ファミリ タイプの情報を表示するために、同じ関数を使用することもできます。

```

<C#>
    // (3) identify each major types of element.
    IdentifyElement(elem);

    // (4) first parameters.
    ShowParameters(elem, "Element Parameters");

    // check to see its type parameter as well
    //
    ElementId elemTypeId = elem.GetTypeId();
    ElementType elemType = (ElementType)m_rvtDoc.GetElement(elemTypeId);
    ShowParameters(elemType, "Type Parameters");
</C#>

```

“DB Element” コマンドをビルドして、再度実行してみてください。ダイアログにパラメータの一覧が表示されるはずです。P2の図2は、コマンド実行時のサンプル画像を示しています。

5.2 組み込みパラメータ(BuiltInParameter)を使った個別パラメータの取得

個々のパラメータにアクセスするには、4つの方法があります：

- Parameter(**BuiltInParameter**) — 組み込みパラメータ 列挙タイプの値 を使用してパラメータを取得する
- LookupParameter (String) — 名前を使用して取得する
- Parameter(Definition) — その定義から取得する
- Parameter(GUID) — 共有パラメータのGUIDを使用して共有パラメータを取得する

ここでは、1 番目と 2 番目を見てみましょう。名前を使用して Parameter(Xxx) メソッド自体を呼ぶことは簡単です。しかしながら、名前の使用には実行している Revit の言語バージョンに依存するという不利な点があります。したがって、組み込みパラメータの使用がより理想的です。ここでの課題は、組み込みパラメータの Id をどのように見つけ出すかという点です。RevitAPI.chm ドキュメントを見れば、膨大な数の BuiltInParameter 列挙タイプの値を見つけることができます。それらの組み込みパラメータの中から一意の要素に適用可能な値は限られています。特定のパラメータを取得するために、どの組み込みパラメータを使用すべきかどうやって知ればよいでしょうか？

RevitLookup ツールは、どの組み込みパラメータがどのパラメータに相当するか調査するのに役立ちます。特定タイプの要素のパラメータを見付け出したい場合には、プロジェクトか特定のタイプのオブジェクトを選択して、[Snoop Current Selection ...] >> [Parameters] をクリックします。各パラメータ名をクリックすれば、それに対応する組み込みパラメータを見付けることができます(図 5)。

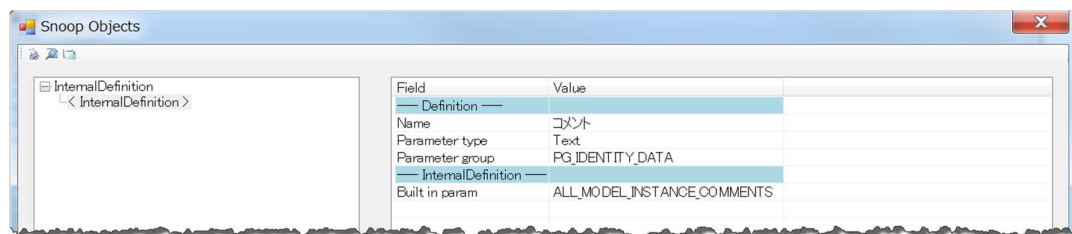


図 5.パラメータの定義をチェックするために RevitLookup を使用

組み込みパラメータの一覧を参照する場合は[Built-in Enums Snoop...] ボタンや [Built-in Enums Map...] ボタンを使って参照できます。例えば、次の組み込みパラメータを見付けることができます。

- BuiltInParameter.SYMBOL_FAMILY_AND_TYPE_NAMES_PARAM — ファミリとタイプ名
- BuiltInParameter.SYMBOL_FAMILY_NAME_PARAM — ファミリ名

要素のファミリおよびタイプ名を取得するために、これらを使用することができます。図 6 は、組み込みパラメータ列挙タイプとパラメータのマッピングを示します。

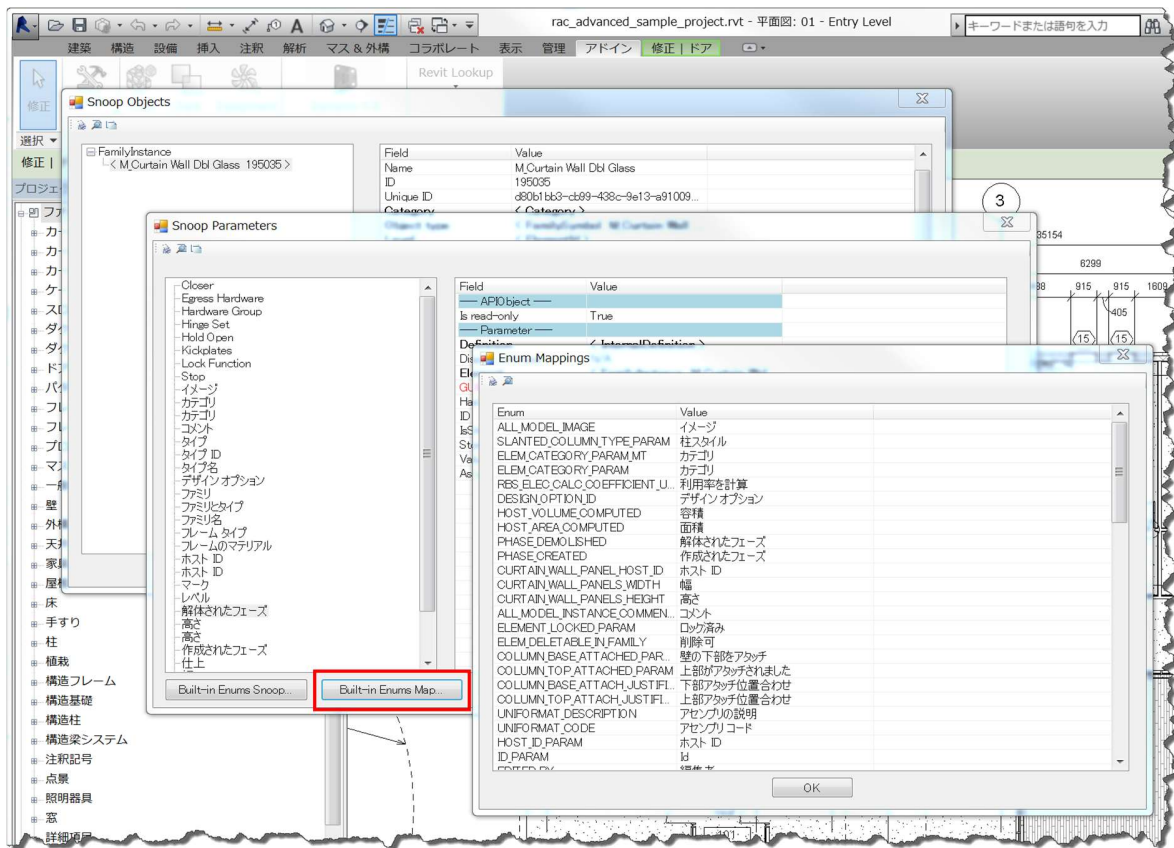


図 6.どの組み込みパラメータを使用すべきか調査するために RevitLookup ツールを使用した例

下記は一般的に使われるプロパティのいくつかを取得する方法を示すサンプルです。RevitLookup ツール内で組み込みパラメータを調査して下記のように記述してください。

```
<C#>
// example of retrieving a specific parameter indivisually.
// (hard coding for simplicity. This function works best
// with walls and doors.)

public void RetrieveParameter(Element elem, string header)
{
    string s = string.Empty;

    // as an experiment, let's pick up some arbitrary parameters.
    // comments - most of instance has this parameter

    // (1) by BuiltInParameter.
    Parameter param =
        elem.get_Parameter(BuiltInParameter.ALL_MODEL_INSTANCE_COMMENTS);
```

```

if (param != null)
{
    s += "Comments (by BuiltInParameter) = " +
        ParameterToString(param) + "\n";
}

// (2) by name. (Mark - most of instance has this parameter.)
// if you use this method, it will language specific.
param = elem.LookupParameter("マーク"); // Revit 2015
if (param != null)
{
    s += "Mark (by Name) = " + ParameterToString(param) + "\n";
}

// the following should be in most of type parameter
//
param = elem.get_Parameter(BuiltInParameter.ALL_MODEL_TYPE_COMMENTS);
if (param != null)
{
    s += "Type Comments (by BuiltInParameter) = " +
        ParameterToString(param) + "\n";
}

param = elem.LookupParameter("耐火等級"); // Revit 2015
if (param != null)
{
    s += "Fire Rating (by Name) = " + ParameterToString(param) +
        "\n";
}

// using the BuiltInParameter, you can sometimes access one that is
// not in the parameters set.
// Note: this works only for element type.

param = elem.get_Parameter(
    BuiltInParameter.SYMBOL_FAMILY_AND_TYPE_NAMES_PARAM);
if (param != null)
{
    s += "SYMBOL_FAMILY_AND_TYPE_NAMES_PARAM (only by
BuiltInParameter) = " +
        ParameterToString(param) + "\n";
}

param = elem.get_Parameter(BuiltInParameter.SYMBOL_FAMILY_NAME_PARAM);
if (param != null)
{
    s += "SYMBOL_FAMILY_NAME_PARAM (only by BuiltInParameter) = "
        + ParameterToString(param) + "\n";
}

// show it.

TaskDialog.Show(header, s);

```

```
}  
</C#>
```

Execute () メソッドの終わりで、この関数を呼び出します。さらに、ファミリ タイプの情報を表示するために、同じ関数を使用することもできます。

```
<C#>  
// (4) first parameters.  
ShowParameters(elem, "Element Parameters: ");  
  
// check to see its type parameter as well  
//  
ElementId elemTypeId = elem.GetTypeId();  
ElementType elemType = (ElementType)m_rvtDoc.GetElement(elemTypeId);  
ShowParameters(elemType, "Type Parameters: ");  
  
// access to each parameters.  
RetrieveParameter(  
    elem, "Element Parameter (by Name and BuiltInParameter)");  
// the same logic applies to the type parameter.  
RetrieveParameter(  
    elemType, "Type Parameter (by Name and BuiltInParameter)");  
</C#>
```

再度、“DB Element” コマンドをビルドして実行してください。指定したパラメータの一覧がダイアログに表示されるはずです(図7)。

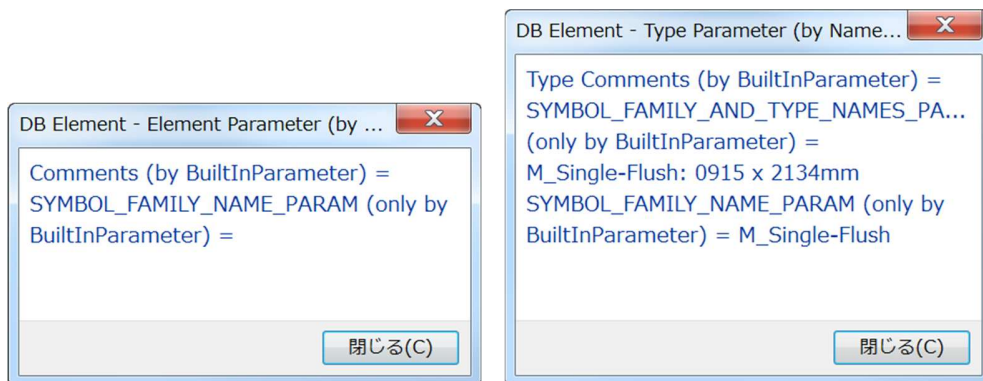


図 7. 個別のパラメータにアクセスするために BuiltInParameters が使用可能

パラメータのより詳細な解説は、Revit API 開発者用ガイドの「[パラメータ](#)」項目をご参照ください。

6. 位置情報

各要素の位置は、Location プロパティに格納されています。位置は、座標ベース(LocationPoint)、あるいは、カーブ/線分ベース(LocationCurve) で取得できます。より多くのプロパティにアクセスするためには、LocationPoint または LocationCurve にキャストする必要があります。下記は、その使用法を示します:

```
<C#>
// show the location information of the given element.
// location can be LocationPoint (e.g., furniture), and LocationCurve
// (e.g., wall).

public void ShowLocation(Element elem)
{
    string s = "Location Information: " + "\n" + "\n";
    Location loc = elem.Location;

    if (loc is LocationPoint)
    {
        // (1) we have a location point

        LocationPoint locPoint = (LocationPoint)loc;
        XYZ pt = locPoint.Point;
        double r = locPoint.Rotation;

        s += "LocationPoint" + "\n";
        s += "Point = " + PointToString(pt) + "\n";
        s += "Rotation = " + r.ToString() + "\n";
    }
    else if (loc is LocationCurve)
    {
        // (2) we have a location curve

        LocationCurve locCurve = (LocationCurve)loc;
        Curve crv = locCurve.Curve;

        s += "LocationCurve" + "\n";
        s += "EndPoint(0)/Start Point = " +
            PointToString(crv.GetEndPoint(0)) + "\n";
        s += "EndPoint(1)/End point = " +
            PointToString(crv.GetEndPoint(1)) + "\n";
        s += "Length = " + crv.Length.ToString() + "\n";

        // Location Curve also has property JoinType at the end

        s += "JoinType(0) = " + locCurve.get_JoinType(0).ToString() + "\n";
        s += "JoinType(1) = " + locCurve.get_JoinType(1).ToString() + "\n";
    }

    // show it.
}
```

```

        TaskDialog.Show("Show Location", s);
    }

    // Helper Function: returns XYZ in a string form.
    //
    public static string PointToString(XYZ pt)
    {
        if (pt == null)
        {
            return "";
        }

        return string.Format("{0},{1},{2}",
            pt.X.ToString("F2"), pt.Y.ToString("F2"), pt.Z.ToString("F2"));
    }
}
</C#>

```

Execute () メソッドの終わりで関数を呼び出すようにします。

```

<C#>
    // (4) first parameters.
    ...
    // the same logic applies to the type parameter.
    RetrieveParameter(elemType, "Type Parameter (by Name and
BuiltInParameter): ");

    // (5) location
    ShowLocation(elem);
</C#>

```

再度、“DB Element” コマンドをビルドして実行してください。位置情報がダイアログで表示されるはずです。図 8 では、ドアのような Location Point ベースの要素を選択した際の位置情報の表示例を示しています。壁を選択した際には、Location Line になるはずです。すべての要素が、位置情報へのアクセスが可能とは限らない点に注意してください。

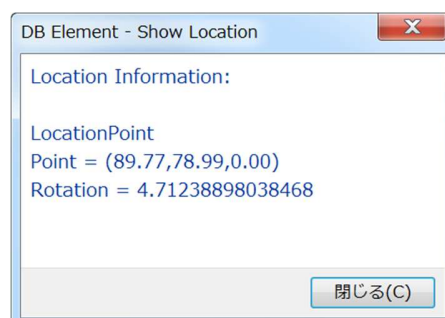


図 8. サンプル位置情報

7. ジオメトリ情報(オプション)

要素プロパティからジオメトリ情報を取得できます。ジオメトリ情報の取得のためにコードを記述するのは、この基礎トレーニングの範囲を超えてしまうので、ここでは簡単な紹介にとどめます。

- ジオメトリ・オプション — 要素からジオメトリ情報を取得する場合、詳細レベル(簡略、標準、詳細)を指定することができます。
- ジオメトリ・オブジェクトにはソリッド、曲線、メッシュまたはそれらを含むジオメトリ・インスタンスなどが存在します。

下記のコードは、ジオメトリ情報にアクセスする簡単な一例です。ソリッド/面/エッジなどより詳しい探索のためには、RevitLookup ツールを使用することもできます。Revit SDKに含まれるRevitCommands サンプルはより詳細にジオメトリ情報にアクセスしています。また次のSDKサンプルは要素のジオメトリ情報を取得してそれらを簡単なビューアで描画します。

- ElementViewer
- RoomViewer
- AnalyticalViewer

```
<C#>
// show the geometry information of the given element.
public void ShowGeometry(Element elem)
{
    // Set a geometry option
    Options opt = m_rvtApp.Create.NewGeometryOptions();
    opt.DetailLevel = ViewDetailLevel.Fine;

    // Get the geometry from the element
    GeometryElement geomElem = elem.get_Geometry(opt);

    // If there is a geometry data, retrieve it as a string to show it.
    string s = (geomElem == null) ?
        "no data" :
        GeometryElementToString(geomElem);

    TaskDialog.Show("Show Geometry", s);
}

// Helper Function: parse the geometry element by geometry type.
// Here we look at the top level.

public static string GeometryElementToString(GeometryElement geomElem)
{
    string str = string.Empty;

    foreach (GeometryObject geomObj in geomElem)
    {
        if (geomObj is Solid)
```

```

    {
        // ex. wall

        Solid solid = (Solid)geomObj;
        //str += GeometrySolidToString(solid)

        str += "Solid" + "\n";
    }
    else if (geomObj is GeometryInstance)
    {
        // ex. door/window

        str += " -- Geometry.Instance -- " + "\n";
        GeometryInstance geomInstance = (GeometryInstance)geomObj;
        GeometryElement geoElem = geomInstance.SymbolGeometry;

        str += GeometryElementToString(geoElem);
    }
    else if (geomObj is Curve)
    {
        Curve curv = (Curve)geomObj;
        //str += GeometryCurveToString(curv)

        str += "Curve" + "\n";
    }
    else if (geomObj is Mesh)
    {
        Mesh mesh = (Mesh)geomObj;
        //str += GeometryMeshToString(mesh)

        str += "Mesh" + "\n";
    }
    else
    {
        str += " *** unkown geometry type " +
            geomObj.GetType().Name;
    }
}

return str;
}
</C#>

```

Execute () メソッドの終わりで関数を呼び出すようにします。

```

<C#>
    // (5) location
    ShowLocation(elem);

    // (6) geometry - the last piece. (Optional)
    ShowGeometry(elem);
</C#>

```

“DB Element” コマンドをビルドして実行してください。ジオメトリ情報がダイアログで表示されるはずです(図9)。

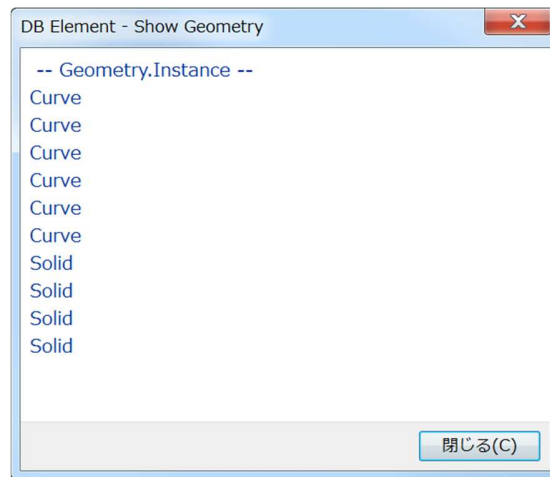


図9.ジオメトリ情報の例

8. サマリ

この実習では、Revit 内で要素がどのように表現されているか、また、要素に関する情報の取得方法について学習しました。習得したのは、次の項目です。

- クラス名、カテゴリおよびシンボルを使用して要素を識別する
- Parameters ()を使用して要素のプロパティ セットを取得する
- BuiltInParameter を使用して要素の特定のプロパティを取得する
- 位置情報を取得する
- ジオメトリ情報を取得する

次の実習では、Revit データベース内の目的の要素を取得する方法を学習します。その機能は、要素フィルタリングと呼ばれるものです。