

Revit API トレーニング

Developer Technical Services



アジェンダ

- Day 1
 - API カスタマイズ概要と Revit データベースの基礎
 - コマンドとアプリケーション
 - 開発ツール
 - Revit 要素とクラス ...
- Day 2
 - Revit ユーザ インタフェースの基礎
 - ファミリ API
 - その他のトピック

トレーニング マテリアルに関する注意事項

- C ドライブに “Revit API Training” フォルダごとコピー
 - C:¥Revit API Training
- 実習用のプロジェクト作成
 - 標準の「建設テンプレート」(Construction-DefaultJPNJPN.rte) でプロジェクトを新規作成して下さい
 - ファミリ API 実習用は「柱(メートル単位).rft」標準ファミリ テンプレートでファミリを新規作成して下さい

API カスタマイズ概要と Revit データベースの基礎



アジェンダ

- Revit API の概要
 - 製品構成
 - Revit SDK
 - ドキュメントとサンプル
- 開発環境と Revit アドイン
 - 外部コマンドと 外部アプリケーション — HelloWorld
 - アドイン マニフェスト、RvtSamples と RevitLookup
- データベースの基礎
 - Revit 要素の理解と表現
 - 要素の走査、フィルタリングとクエリ
 - 要素の修正
 - モデル作成

Revit API カスタマイズ概要

製品、SDK、アセンブリ DLL

Revit の製品構成

- 4 タイプの Revit 製品
 - Revit Architecture
 - Revit MEP
 - Revit Structure
 - Onebox
 - Building Design Suiteに含まれる Revit
- 製品ビルドの入手方法
 - ADN Extranet(adn.autodesk.com)
 - ソフトウェア リクエスト フォームで申請
 - www.autodesk.co.jp
 - 体験版をダウンロード

Revit API のテクノロジーと提供方法

- Revit API
 - .NET Framework ベース テクノロジーを採用
 - Revit API は .NET アセンブリで公開
- Revit API アセンブリの提供場所
 - Revit のインストールフォルダ直下
- DB と UI を分離してアセンブリを提供
 - RevitAPI.dll
 - RevitAPIUI.dll
- Revit Architecture、Structure、MEP の API 差異
 - 共通化されたアセンブリ DLL で提供
 - 一部機能は各製品内でのみで利用可能

Revit SDKの入手

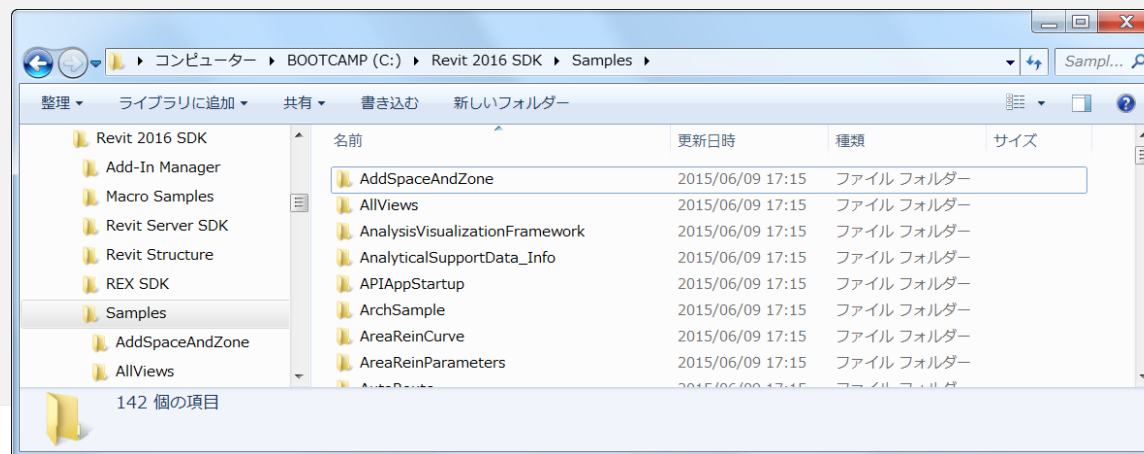
- 製品インストーラに同梱
 - ツールとユーティリティ インストール
 - Revit Software Development Kit
- Web ダウンロードの場合
 - <展開フォルダ>¥Utilities¥SDK¥RevitSDK.exe
- 最新バージョンのSDK ダウンロード先（推奨）
 - <http://www.autodesk.com/developrevit> または
 - <http://www.autodesk.co.jp/developrevit>

Revit SDK

- 既定インストール フォルダ
 - C:¥Revit 20xx SDK
- ドキュメント(英語)
 - はじめにお読みください: Read Me First.doc
 - リファレンス マニュアル: RevitAPI.chm
- ツール
 - Add-In Manager
 - RevitLookup
- サンプル
 - 機能別の各種サンプル プロジェクト

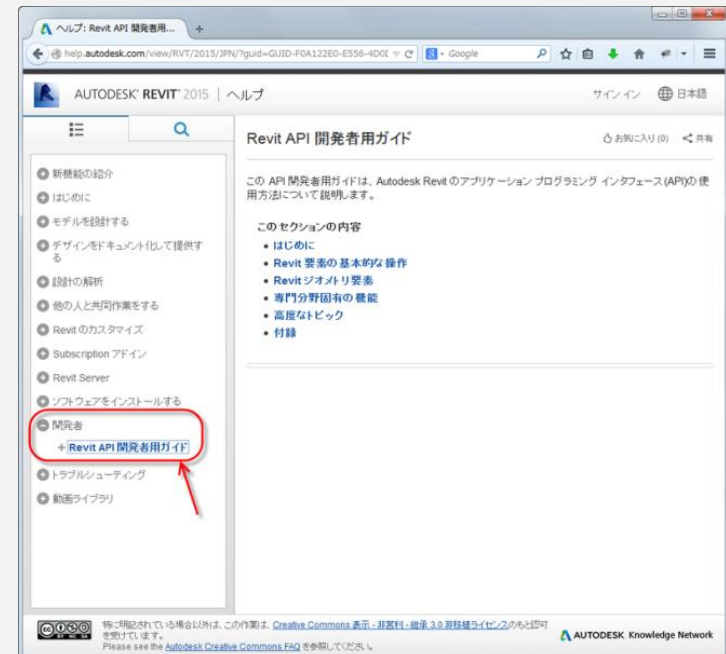
Revit SDK サンプル

- サンプル概要の把握
 - SamplesReadMe.htm
- ユーティリティ
 - RevitAPIDllsPathUpdater.exe
 - サンプル プロジェクトのアセンブリ参照情報を一括更新
- サンプル ソリューション
 - SDKSamples20xx.sln
 - Visual Studio 20xx 用ソリューション ファイル



Revit API 開発者用ガイド

- Revit API 開発者用ガイドは Revit 製品ヘルプに含まれています
- Revit 製品ヘルプ オンライン版からご利用できます
 - <http://help.autodesk.com/view/RVT/20xx/JPN/>



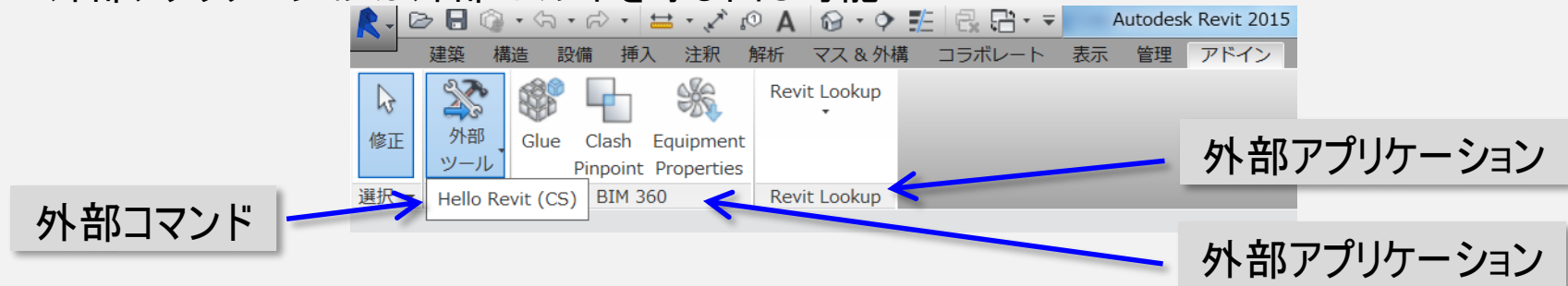
API を利用した Revit の拡張手法

1. 外部コマンド

- IExternalCommand インタフェースを利用する実装; アドイン マニフェストをインストール
- コマンドは [アドイン] リボンタブの [外部ツール] に追加される

2. 外部アプリケーション

- IExternalApplication インタフェースを利用する実装; アドイン マニフェストをインストール
- アプリケーションは [アドイン] リボンタブに新しいパネルを作成することが可能
- 外部アプリケーションは外部コマンドを呼び出し可能



(BIM 360 ツールは外部アプリケーションとして Revit 2015に追加されています)

3. SharpDevelop マクロ

- 従来の VSTA の替わるオープンソース開発環境(今回のトレーニングではカバーしません)

Revit API を利用したアドインの開発環境

- 開発環境 (Revit 2017, Revit 2018 対応環境)
 - Microsoft Visual Studio 2015
 - 開発言語
 - C#、VB.NET、マネージ C++、その他 .NET 準拠の言語
 - .NET Framework 4.5.2
 - サポート OS
 - Windows 7 (x86、x64)、Windows 8 (x64)
 - Windows XP、Vista と Windows 8 (x86) はサポートなし
- 参照設定が必要なアセンブリ
 - <Revit インストール フォルダ>\¥RevitAPI.dll
 - <Revit インストール フォルダ>\¥RevitAPIUI.dll
 - ※ アセンブリの 'ローカル コピー' は False に設定が必要

外部コマンド

Hello World の作成 : 外部コマンドとアドイン マニフェスト

Hello World 外部コマンドの作成手順

1. クラス ライブラリ プロジェクトの新規作成
2. 必要となる参照設定(最小):
 - System.dll
 - RevitAPI.dll
 - RevitAPIUI.dll
3. 通常使用する名前空間のインポート
 - Autodesk.Revit.DB
 - Autodesk.Revit.UI
 - Autodesk.Revit.ApplicationServices
 - Autodesk.Revit.Attributes
4. IExternalCommand 派生クラスと Execute() メソッドを実装
5. アドイン マニフェストの作成とインストール

外部コマンドの実装

VB.NET での最小コード

```
<VB.NET>
'' Hello World #1 - A minimum Revit external command.
<Autodesk.Revit.Attributes.Transaction(Autodesk.Revit.Attributes.TransactionMode.Manual)> _
Public Class HelloWorld
    Implements IExternalCommand

    Public Function Execute( _
        ByVal commandData As Autodesk.Revit.UI.ExternalCommandData, _
        ByRef message As String, _
        ByVal elements As Autodesk.Revit.DB.ElementSet) _

        As Autodesk.Revit.UI.Result _
        Implements Autodesk.Revit.UI.IExternalCommand.Execute

        Autodesk.Revit.UI.TaskDialog.Show("My Dialog Title", "Hello World!")

        Return Result.Succeeded

    End Function

End Class
</VB.NET>
```

外部コマンドの実装

C# での最小コード

```
<C#>
// Hello World #1 - A minimum Revit external command.
[Autodesk.Revit.Attributes.Transaction(Autodesk.Revit.Attributes.TransactionMode.Manual)]
public class HelloWorld : IExternalCommand
{
    public Autodesk.Revit.UI.Result Execute(
        Autodesk.Revit.UI.ExternalCommandData commandData,
        ref string message,
        Autodesk.Revit.DB.ElementSet elements)
    {
        Autodesk.Revit.UI.TaskDialog.Show("My Dialog Title", "Hello World!");

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

外部コマンドの実装

IExternalCommand クラス

<VB.NET>

```
' Hello World #1 - A minimum Revit external command.  
<Autodesk.Revit.Attributes.Transaction(Autodesk.Revit.Attributes.TransactionMode.Automatic)> _
```

```
Public Class HelloWorld  
    Implements IExternalCommand
```

```
    Public Function Execute( _  
        ByVal commandData As Autodesk.Revit.UI.ExternalCommandData, _  
        ByRef message As String, _  
        ByVal elements As Autodesk.Revit.DB.ElementSet) _  
        As Autodesk.Revit.UI.Result _  
        Implements Autodesk.Revit.UI.IExternalCommand.Execute  
  
        Autodesk.Revit.UI.TaskDialog.Show("My Dialog Title", "Hello World!")  
  
        Return Result.Succeeded  
  
    End Function
```

```
End Class
```

</VB.NET>

1. IExternalCommand からのクラス派生

外部コマンドの実装

Execute() メソッド

<VB.NET>

```
' Hello World #1 - A minimum Revit external command.  
<Autodesk.Revit.Attributes.Transaction(Autodesk.Revit.Attributes.TransactionModeAutomatic)> _  
Public Class HelloWorld  
    Implements IExternalCommand
```

2.Execute() メソッドの実装

```
Public Function Execute( _  
    ByVal commandData As Autodesk.Revit.UI.ExternalCommandData, _  
    ByRef message As String, _  
    ByVal elements As Autodesk.Revit.DB.ElementSet) _  
  
    As Autodesk.Revit.UI.Result _  
    Implements Autodesk.Revit.UI.IExternalCommand  
  
    Autodesk.Revit.UI.TaskDialog.Show("My Dialog")  
  
    Return Result.Succeeded  
  
End Function
```

戻り値:

- Succeeded
- Failed
- Cancelled

パラメータ:

- 1st Access to the Revit object model
- 2nd Message to the user when a command fails
- 3rd A set of elements to be highlighted when a command fails

```
End Class  
</VB.NET>
```

外部コマンドの実装

属性

<VB.NET>

```
' Hello World #1 - A minimum Revit external command.  
<Autodesk.Revit.Attributes.Transaction(Autodesk.Revit.Attributes.TransactionMode.Manual)> _  
Public Class HelloWorld  
    Implements IExternalCommand  
  
    Public Function Execute(  
        ByVal command As IExternalCommandData, _  
        ByRef message As String, _  
        ByVal elements As IEnumerable<IElement> _  
        As Autodesk.Revit.UI.Result _  
        Implements Autodesk.Revit.UI.IExternalCommand.Execute  
  
        Autodesk.Revit.UI.TaskDialog.Show("My Dialog Title", "Hello World!")  
  
        Return Result.Succeeded  
  
    End Function  
  
End Class  
</VB.NET>
```

3. 属性の設定

トランザクション モード: トランザクション動作をコントロール

- Manual
- ReadOnly

外部コマンドの実装

Hello World の表示

<VB.NET>

```
' Hello World #1 - A minimum Revit external command.
<Autodesk.Revit.Attributes.Transaction(Autodesk.Revit.Attributes.TransactionModeAutomatic)> _
Public Class HelloWorld
    Implements IExternalCommand

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

        Autodesk.Revit.UI.TaskDialog.Show("My Dialog Title", "Hello World!")

        Return Result.Succeeded

    End Function

End Class
</VB.NET>
```

4. メッセージをダイアログで表示

タスク ダイアログ：
Revit スタイルのメッセージボックス
“Hello World” を表示

アドイン マニフェスト

登録メカニズム

- 起動時に Revit によって自動的に読み込まれます
- 用途に応じて 2 つのパスを使い分け：
 1. すべてのユーザ

Windows 7 / Windows 8

C:\ProgramData\Autodesk\Revit\Addins\20xx

2. ログインユーザ毎

Windows 7/ Windows 8

C:\Users\<user>\AppData\Roaming\Autodesk\Revit\Addins\20xx

アドイン マニフェスト

.addin ファイル

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<RevitAddIns>
  <AddIn Type="Command">
    <Assembly>C:\¥...¥HelloWorld.dll</Assembly>
    <FullName>IntroVb.HelloWorld</FullName>
    <Text>Hello World</Text>
    <AddInId>0B997216-52F3-412a-8A97-58558DC62D1E</AddInId>
    <VendorId>ADNP</VendorId>
    <VendorDescription>Autodesk, Inc. www.autodesk.com</VendorDescription>
  </AddIn>
</RevitAddIns>
```

マニフェスト内の情報:

- アドイン タイプ: コマンド または アプリケーション
- Text は [アドイン] タブの [外部ツール] パネルに表示
- 名前空間を含む完全なクラス名
- クラス名は **大文字小文字を区別**
- アセンブリ ファイル (dll) へのフルパス
- コマンドの GUID ないし一意な識別子

➤ 詳細は Revit API 開発者用ガイド
「アドインの登録」を参照

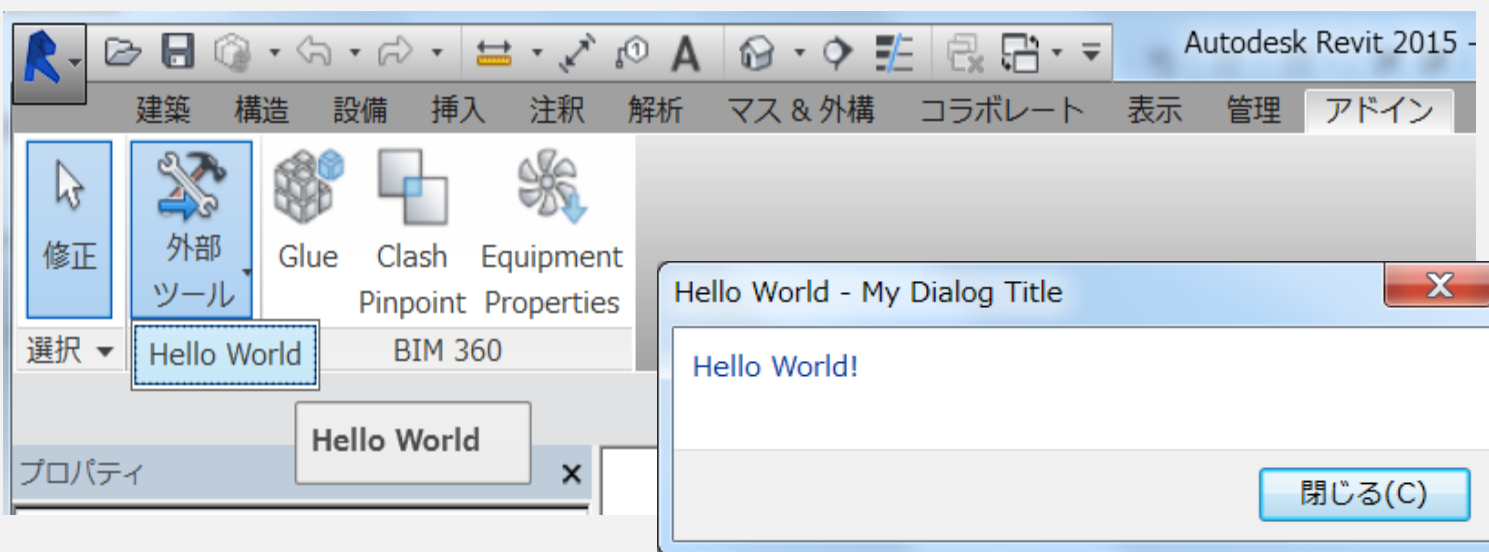
Vendor Id 用の Registered Developer Symbol

- Vendor Id は一意であるべき
- 一意なシンボル取得の方法:
 - オートデスク **Registered Developer Symbol (RDS)**
 - <http://www.autodesk.com/symbreg/index.htm> または
“autodesk register developer symbol” で Google 検索
- Autodesk Developer Center でシンボルを登録
 - 4文字の半角英数字
 - 使用不可の文字: % . @ * [] { } ^ \$ / ¥ と特殊文字
- 全ての ADN プラグインは “ADN Plugin” 用に “ADNP” を使用

外部ツール パネル

アドインの実行

- マニフェストによる読み込みで [アドイン] タブの[外部ツール] に表示
- プルダウン メニューからコマンドを実行



外部アプリケーション

外部アプリケーションと外部コマンドのデータ

外部アプリケーション

VB.NET での最小コード

<VB.NET>

' Hello World App - minimum external application

Public Class HelloWorldApp

Implements IExternalApplication

IExternalApplication を実装

' OnShutdown() - called when Revit ends.

Public Function OnShutdown(ByVal application As UIControlledApplication) _
As Result _

Implements IExternalApplication.OnShutdown

Return Result.Succeeded

End Function

Revit 終了時に OnShutdown を呼び出し

' OnStartup() - called when Revit starts.

Public Function OnStartup(ByVal application As UIControlledApplication) _
As Result _

Implements IExternalApplication.OnStartup

TaskDialog.Show("My Dialog Title", "Hello World from App!")

Return Result.Succeeded

End Function

Revit 起動時に OnShutdown を呼び出し

End Class

</VB.NET>

外部アプリケーション

C# での最小コード

<C#>

```
// Hello World #3 - minimum external application
public class HelloWorldApp : IExternalApplication
{
    // OnStartup() - called when Revit starts.
    public Result OnStartup(UIControlledApplication application)
    {
        TaskDialog.Show("My Dialog Title", "Hello World from App!");
        return Result.Succeeded;
    }

    // OnShutdown() - called when Revit ends.
    public Result OnShutdown(UIControlledApplication application)
    {
        return Result.Succeeded;
    }
}
```

</C#>

外部アプリケーション

アドイン マニフェスト(.addin)

Type = “Command” の代わりに “Application”
<Text> の代わりに <Name> を使用

```
<AddIn Type="Application">  
  <Name>Hello World App</Name>  
  <FullClassName>IntroVb.HelloWorldApp</FullClassName>  
  <Assembly>C:¥...¥IntroVb.dll</Assembly>  
  <AddInId>08E8EFB1-FCC1-4b99-AD14-93523EE229AA</AddInId>  
  <VendorId>ADNP</VendorId>  
  <VendorDescription>Autodesk, Inc. www.autodesk.com</VendorDescription>  
</AddIn>
```

Revit オブジェクト モデルへのアクセス

- ## ■ Revit モデルにアクセスするためのオブジェクト

```
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.
    ' Notice that we have UI and DB versions for application and Document.
    ' (We list them both here to show two versions.)

    Dim uiApp As UIApplication = commandData.Application
    Dim uiDoc As UIDocument = uiApp.ActiveUIDocument
    _app = uiApp.Application
    _doc = uiDoc.Document

    ' (1) select an object on a screen. (We'll select the first object.)
    Dim ref As Reference = _doc.Selection.PickObject(ObjectType.Element)

    ' We have picked something.
    Dim e As Element = _doc.GetElement(ref)

    ' (2) let's see what kind of element we have
    ' Key properties that we need to check are:
    ' - Element type
    ' - Element name
    ' - Element parameters

    ShowBasicElementInfo(e)

    ' (3) now, we are going to identify each element
    IdentifyElement(e)

    ' Now look at other properties - important ones
    ' (4) first parameters.

    ShowParameters(e, "Element Parameters")

    ' Check to see its type parameter as well.
```

コマンド データ

Revit オブジェクト モデルへのアクセス

例:

```
" Revit バージョンと使用中のドキュメント タイトルの取得
```

```
Dim versionName As String = _  
    commandData.Application.Application.VersionName  
Dim documentTitle As String = _  
    commandData.Application.ActiveUIDocument.Document.Title
```

```
" 現在の RVT プロジェクトで利用可能な壁タイプを表示
```

```
Dim wallTypes As WallTypeSet = _  
    commandData.Application.ActiveUIDocument.Document.WallTypes  
Dim s As String = ""  
    For Each wType As WallType In wallTypes  
        s = s + wType.Name + vbCr  
    Next
```


コマンド データ

Revit オブジェクト モデルへのアクセス

- DB と UI 部分での アプリケーションと ドキュメント へのアクセス

<VB.NET>

```
Public Class DBElement
    Implements IExternalCommand

    ' ' member variables
    Dim m_rvtApp As Application
    Dim m_rvtDoc As Document

    Public Function Execute(ByVal commandData As ExternalCommandData, _
        ...

        ' ' 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

        ' ' ...
```

</VB.NET>

ツール

Revit Lookup、Add-In Manager、SDKSamples2016.sln、RvtSamples

ツール

必須知識

■ RevitLookup

- Revit データベース構造を調査する Revit API プログラマ必須ツール

■ Add-In Manager

- 起動中の Revit にアドインをロード/ロード解除させるツール
- アドイン マニフェストの登録なしで外部コマンドの実行が可能

■ SDKSamples20xx.sln

- サンプル プロジェクト群を一括してビルドするための VS ソリューション
- **RevitAPIDllsPathUpdater.exe** でサンプル プロジェクト群が参照するアセンブリの参照設定を VS で各プロジェクトを開かずに一括設定

■ RvtSamples

- 全サンプルのテスト用にリボンパネルを作成するサンプル プロジェクト

DB 要素

Revit 要素の表現を理解

DB 要素(エレメント)

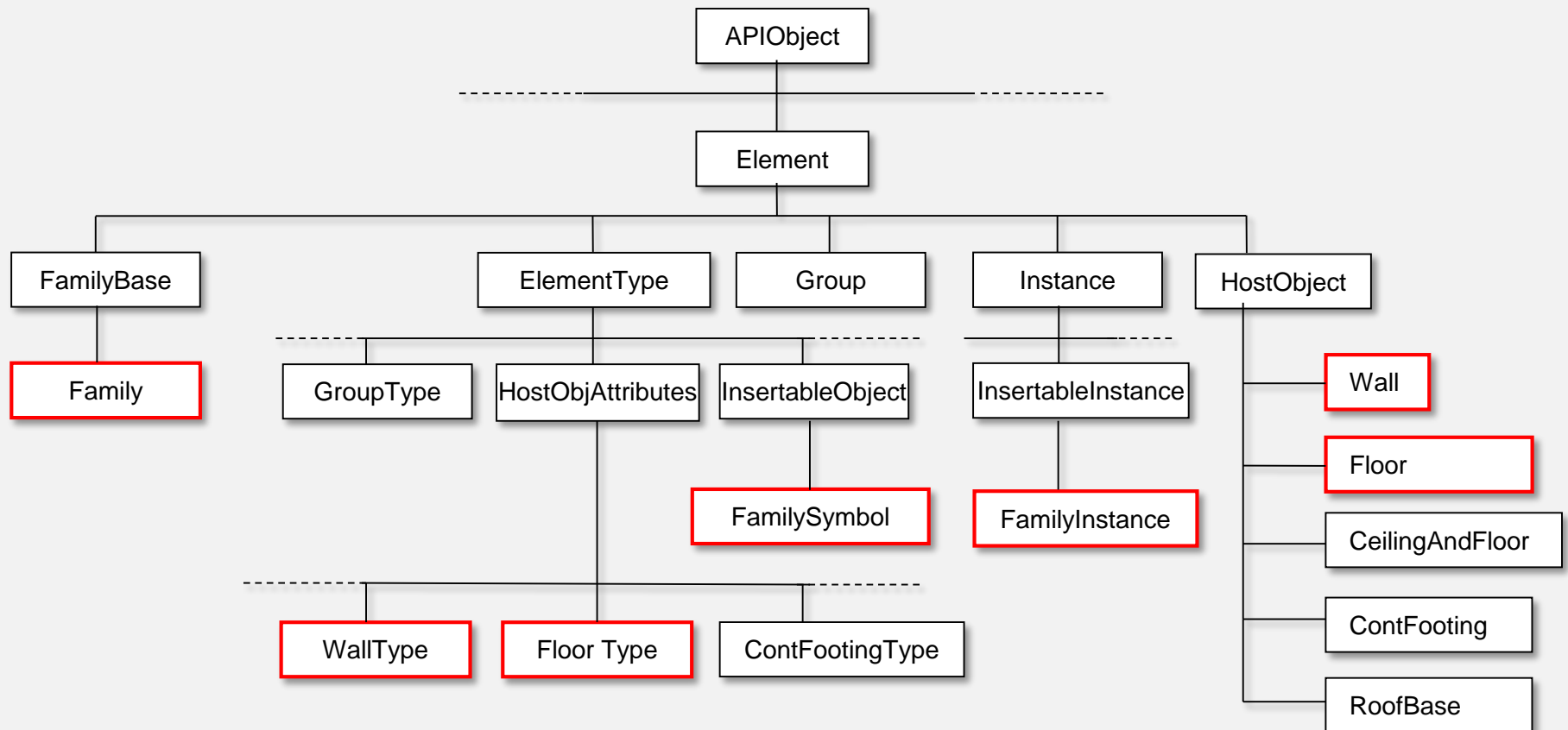
要素の基本

Revit API は通常のプログラミングのようにクラス名を
チェックしてオブジェクトを識別するのですか？

いいえ

DB 要素

クラス派生



シンボルと呼ばれるファミリとタイプ

標準とシステムのホストとコンポーネント オブジェクト

DB 要素

要素とシンボル

要素			シンボル		
UI の要素種別	要素/タイプ別から派生	カテゴリ	要素/タイプ別から派生	カテゴリ	
壁	HostObject/Wall	壁	HostObjAttributes/WallType	壁	
ドア	Instance/InsertableInstance/ FamilyInstance	ドア	InsertableObject /FamilySymbol	ドア	
ドアタグ	IndependentTag	ドアタグ	InsertableObject /FamilySymbol	ドアタグ	
窓	Instance/InsertableInstance/ FamilyInstance	窓	InsertableObject /FamilySymbol	窓	
窓タグ	IndependentTag	窓タグ	InsertableObject /FamilySymbol	窓タグ	
開口部	Opening	開口部	< null >	---	
床	HostObject/CeilingAndFloor /Floor	床	HostObjAttributes/FloorType	床	
天井	HostObject/CeilingAndFloor /Ceiling	天井	HostObjAttributes/CeilingType	天井	
屋根	HostObject/RoofBase/FootPrintRoof, ExtrusionRoof	屋根	HostObjAttributes/RoofType	屋根	
柱	Instance/InsertableInstance/ FamilyInstance	柱	InsertableObject /FamilySymbol	柱	
机	Instance/InsertableInstance/ FamilyInstance	家具	InsertableObject /FamilySymbol	家具	
木	Instance/InsertableInstance/ FamilyInstance	植栽	InsertableObject /FamilySymbol	植栽	
階段	Stairs	階段	< Symbol >	階段	
手すり	Railing	手すり	< Symbol >	手すり	
部屋	Room	部屋	< null >	---	
部屋タグ	RoomTag	部屋タグ	< Symbol >	部屋タグ	
グリッド	Grid	グリッド	LineAndTextAttrSymbol/GridType	< null >	
モデル線分	ModelCurve/ModelLine	線分	< null >	---	
参照面	ReferencePlane	参照面	< null >	---	
寸法	Dimension	寸法	DimensionType	< null >	
断面	< Element >	ビュー	< Symbol >	< null >	
文字	TextElement/TextNote	文字	TextElementType/TextNoteType	< null >	
レベル	Level	レベル	LevelType	レベル	
モデルグループ	Group	モデルグループ	GroupType	モデルグループ	
インプレースを作成					
/壁	Instance/InsertableInstance/ FamilyInstance	壁	InsertableObject /FamilySymbol	壁	

DB 要素

要素の識別

- システム ファミリ
 - Revit 組み込みオブジェクト
 - 各オブジェクト用のクラスが存在
 - 要素を識別する目的で使用可能
- コンポーネント ファミリ
 - FamilyInstance/FamilySymbol などからの系統を持つ
 - カテゴリは Revit 内で表現するオブジェクトの種類を識別する方法
- 操作対象の要素に応じたチェックが必要:
 - クラス名
 - カテゴリ プロパティ
 - 要素が Element オブジェクトならタイプがシンボルか否か

	システム ファミリ	コンポーネント ファミリ
ファミリー 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

<VB.NET>

```
' ' identify the type of the element known to the UI.
```

```
Public Sub IdentifyElement(ByVal elem As Element)
```

```
    Dim s As String = ""
```

```
    If TypeOf elem Is Wall Then
```

```
        s = "Wall"
```

```
    ElseIf TypeOf elem Is Floor Then
```

```
        s = "Floor"
```

```
    ElseIf TypeOf elem Is RoofBase Then
```

```
        s = "Roof"
```

```
    ElseIf TypeOf elem Is FamilyInstance Then
```

```
        ' ' An instance of a component family is all FamilyInstance.
```

```
        ' ' We'll need to further check its category.
```

```
        If elem.Category.Id.IntegerValue = _
```

```
            BuiltInCategory.OST_Doors Then
```

```
            s = "Door"
```

```
        ElseIf elem.Category.Id.IntegerValue = _
```

```
            BuiltInCategory.OST_Windows Then
```

```
            s = "Window"
```

```
        ElseIf elem.Category.Id.IntegerValue = _
```

```
            BuiltInCategory.OST_Furniture Then
```

```
            s = "Furniture"
```

```
        Else
```

```
            s = "Component family instance" ' ' e.g. Plant
```

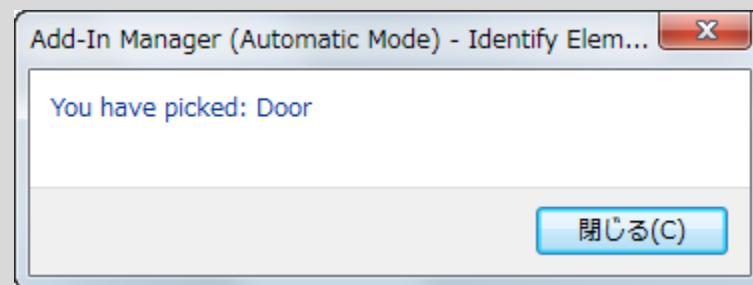
```
        End If
```

```
    ...
```

</VB.NET>

DB 要素

要素の識別



DB 要素

Element.Parameters

- Element クラスの ElementParameters プロパティ
 - UI 内の要素、または、ファミリ “プロパティ” に対応
- API からプロパティやパラメータにアクセスする 2 つの方法:
 - Element.Parameters
 - 与えられた要素に適用可能なパラメータ セットを返す
 - Element.LookupParameter
 - 引数により単一パラメータ値を返す
 - Element.Parameter は同じ名前をもつパラメータが存在することがあるために Revit 2015 で非推奨となりました。

DB 要素

Element.Parameters

```
<VB.NET>
    ' show all the parameter values of the element
    Public Sub ShowParameters(ByVal elem As Element, _
        Optional ByVal header As String = "")

        Dim s As String = header + vbCrLf + vbCrLf
        Dim params As ParameterSet = elem.Parameters

        For Each param As Parameter In params
            Dim name As String = param.Definition.Name
            ' see the helper function below
            Dim val As String = ParameterToString(param)
            s = s + name + " = " + val + vbCrLf
        Next

        TaskDialog.Show("Revit Intro Lab", s)

    End Sub
</VB.NET>
```

<VB.NET>

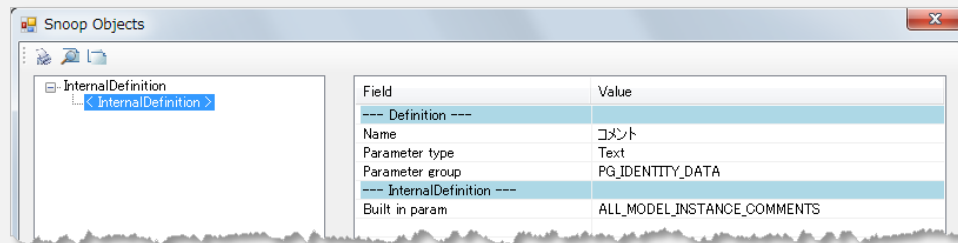
```
' ' Helper function: return a string from of a given parameter.
Public Shared Function ParameterToString(ByVal param As Parameter) As String
    Dim val As String = "none"
    If param Is Nothing Then
        Return val
    End If
    ' ' to get to the parameter value, we need to pause it depending on
    ' ' its strage type
    Select Case param.StorageType
        Case StorageType.Double
            Dim dVal As Double = param.AsDouble
            val = dVal.ToString
        Case StorageType.Integer
            Dim iVal As Integer = param.AsInteger
            val = iVal.ToString()
        Case StorageType.String
            Dim sVal As String = param.AsString
            val = sVal
        Case StorageType.ElementId
            Dim idVal As ElementId = param.AsElementId
            val = idVal.IntegerValue.ToString
        Case StorageType.None
        Case Else
    End Select
    Return val
End Function
```

</VB.NET>

DB 要素

Element.Parameter と組み込みパラメータ

- 個別パラメータにアクセスする 4 つの方法:
 - Parameter(**BuiltInParameter**)
 - パラメータ ID を使ってパラメータを取得
 - LookupParameter(String)
 - パラメータ名を使ってパラメータを取得
 - Parameter(Definition)
 - パラメータの定義からパラメータを取得
 - Parameter(GUID)
 - GUID を使って共有パラメータを取得
- RevitLookup ツールの使用
 - パラメータ名に対応する BuiltInParameterの調査が可能



DB 要素

Element.Parameter と組み込みパラメータ

<VB.NET>

```
' ' examples of retrieving a specific parameter individually.
Public Sub RetrieveParameter(ByVal elem As Element, _
                             Optional ByVal header As String = "")

    Dim s As String = header + vbCrLf + vbCrLf
    ' ' comments - most of instance has this parameter
    ' ' (1) by name. (Mark - most of instance has this parameter.)
    param = elem.LookupParameter("マーク")
    ...
    ' ' (2) by BuiltInParameter.
    Dim param As Parameter = _
        elem.Parameter(BuiltInParameter.ALL_MODEL_INSTANCE_COMMENTS)
    ...
    ' ' using the BuiltInParameter, you can sometimes access one
    ' ' that is not in the parameters set.
    param = elem.Parameter(BuiltInParameter.SYMBOL_FAMILY_AND_TYPE_NAMES_PARAM)
    ...
    param = elem.Parameter(BuiltInParameter.SYMBOL_FAMILY_NAME_PARAM)
    ...
```

</VB.NET>

DB 要素

場所

- Location プロパティ
- Location は2つの形式で派生:
 - LocationPoint – 点ベースの場所(例、家具)
 - LocationCurve – 線分ベースの場所(例、壁)
- より多くのプロパティへアクセス
- LocationPoint または LocationCurve をキャストする必要あり

<VB.NET>

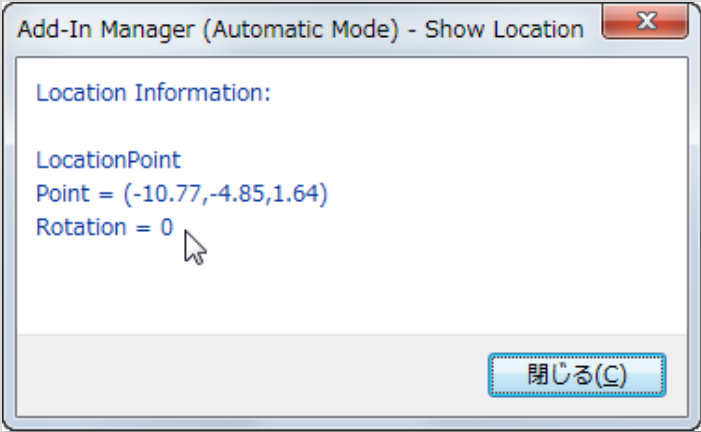
```
' show the location information of the given element.
Public Sub ShowLocation(ByVal elem As Element)
    Dim s As String = "Location Information: " + vbCr + vbCr
    Dim loc As Location = elem.Location

    If TypeOf loc Is LocationPoint Then
        ' (1) we have a location point
        Dim locPoint As LocationPoint = loc
        Dim pt As XYZ = locPoint.Point
        Dim r As Double = locPoint.Rotation
        ...
    ElseIf TypeOf loc Is LocationCurve Then
        ' (2) we have a location curve
        Dim locCurve As LocationCurve = loc
        Dim crv As Curve = locCurve.Curve
        ...

        s = s + "EndPoint(0)/Start Point = " + PointToString(crv.EndPoint(0))
        s = s + "EndPoint(1)/End point = " + PointToString(crv.EndPoint(1))
        s = s + "Length = " + crv.Length.ToString + vbCr
    End If

    ...
End Sub
```

</VB.NET>



DB 要素

ジオメトリ

- ジオメトリ オプション – 詳細レベルを指定
- ジオメトリ オブジェクトの種類
 - Solid
 - Geometry Instance (シンボル要素インスタンス、例、ドア や 窓)
 - Curve
 - Mesh
- Solid/Face/Edge への落とし込み調査 – RevitLookup を使用
- RevitCommands SDK サンプルが単純な例を提供
- SDK サンプルはビューワでジオメトリ アクセスを表示
 - ElementViewer
 - RoomViewer
 - AnalyticalViewer
- その他の表示オプション
 - SVG、VRML、OpenGL、DirectX ... などの利用

要素フィルタ



要素の走査、フィルタリング、照会

要素のフィルタリング

要素の取得

- Revit 内の要素は 1 つに集約されている
- 必要とする要素を取得する一般的な方法：
 1. ファミリ タイプのリストを取得
(例、壁タイプ、ドア タイプ)
 2. 特定のオブジェクト クラスのインスタンスを取得
(例、すべての壁、すべてのドア)
 3. 与えられた名前で特定のファミリ タイプを検索
(例、“標準壁: 標準-200mm”、“片開きフラッシュ: 0915 x 2134mm”)
 4. 特定のインスタンスを検索
(例、“レベル 1”、“平面図 1”)
- 要素がコンポーネント ベースかシステム ベースの考察が必要
 - 要素の識別と同様

FilteredElementCollector

- 要素セットの検索、フィルタ、走査で利用
- 返される要素フィルタへさまざまな条件を割り当てる
- 要素のアクセスを試行する前に最低でも1つの条件が必要
 - そうでない場合は例外をスロー
- IEnumerable インタフェースをサポート
 - マネージ ラッパー生成前にネイティブ コード内で要素を処理するクラスを生成
 - LINQ クエリーを使った試行前にネイティブ フィルタを利用できるため最良のパフォーマンスを得られる

フィルタ タイプ

- 論理フィルタ – フィルタ ロジックの結合が可能
 - And
 - Or
- クイック フィルタ – 遷移状態を判断する内部要素レコードで使用
 - 内部メモリに展開されていない要素を Revit が見つけることを可能に
 - 例:ElementClassFilter
ElementCategoryFilter
- 遅延フィルタ – 要素レコードによる一部の情報の取得が可能
 - フィルタは遷移状態を判断するために展開する必要あり
 - 例:FamilyInstanceFilter
AreaFilter

効率的なガイドライン

- 高速と思われるフィルタを最初に
- 遅いと思われるフィルタを2番目に
- 組み込みフィルタの使用後、他の検索用に LINQ の使用を検討
- FilteredElementCollector のメソッドを使用:
 - 遅延フィルタ用のショートカットが存在しないため
 - クイック フィルタ用の取得ショートカット使用時に確認可能
 - 例:OfClass、OfCategoryId

論理フィルタ

フィルタ名	遷移基準	ショートカット メソッド
LogicalAndFilter	要素は 2つ、または、それ以上の遷移が必要	WherePasses() - 1つの追加フィルタを追加 IntersectWith() - 独立したフィルタの2つの セットを結合
LogicalOrFilter	要素は、少なくとも1つか、または、それ以上の フィルタのどちらかの遷移が必要	UnionWith() - 独立したフィルタの2つの セットを結合

クイック フィルタ

名前	遷移基準	ショートカット メソッド
ElementCategoryFilter	入力したカテゴリ ID と一致する要素	OfCategoryId
ElementClassFilter	入力したランタイム クラスと一致する要素	OfClass
ElementIsElementTypeFilter	“要素タイプ”(シンボル)である要素	WhereElementIsElementType WhereElementIsNotElementType
ElementOwnerViewFilter	ビュー固有の要素	OwnedByView WhereElementIsViewIndependent
ElementDesignOptionFilter	デザイン オプションに含まれる要素	ContainedInDesignOption
ElementIsCurveDrivenFilter	曲線駆動の要素	WhereElementIsCurveDriven
ElementStructuralTypeFilter	与えられた構造タイプに一致する要素	なし
FamilySymbolFilter	ファミリ固有のシンボル	なし
ExclusionFilter	入力したフィルタの要素 ID を除くすべての要素	Excluding
BoundingBoxIntersectsFilter	与えられた外形と交差する境界ボックスを持つ要素	なし
BoundingBoxIsInsideFilter	与えられた外形の内側に境界ボックスを持つ要素	なし
BoundingBoxContainsPointFilter	与えられた点を含む境界ボックスを持つ要素	なし

遅延フィルタ

名前	遷移基準	ショートカット メソッド
FamilyInstanceFilter	ファミリ シンボル固有のインスタンス	なし
ElementLevelFilter	与えられた レベル ID に関連付けられた要素	なし
ElementParameterFilter	パラメータが存在して値と一致、範囲と一致、and/or 文字列と一致する要素	なし
PrimaryDesignOptionMemberFilter	メイン デザイン オプションによって所有される要素	なし
StructuralInstanceUsageFilter	ファミリ インスタンスの構造用途パラメータ	なし
StructuralWallUsageFilter	壁用の構造用途パラメータ	なし
StructuralMaterialTypeFilter	ファミリ インスタンスに適用されるマテリアル タイプ	なし
RoomFilter	部屋の検索	なし
SpaceFilter	スペースの検索	なし
AreaFilter	エリアの検索	なし
RoomTagFilter	部屋のタグの検索	なし
SpaceTagFilter	スペース タグの検索	なし
AreaTagFilter	エリア タグの検索	なし
CurveElementFilter	曲線要素 (モデル曲線、シンボリック曲線、詳細曲線、など) 固有タイプの検索	なし

要素のフィルタリング

1.1 ファミリ タイプのリスト – システム ファミリ

- 全壁タイプの収集 (2つめと3つめはショートカットを使用)

<VB.NET>

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

</VB.NET>

<VB.NET>

```
Dim wallTypeCollector2 = New FilteredElementCollector(m_rvtDoc)
wallTypeCollector2.OfClass(GetType(WallType))
```

</VB.NET>

<VB.NET>

```
Dim wallTypeCollector3 = _
    New FilteredElementCollector(m_rvtDoc).OfClass(GetType(WallType))
```

</VB.NET>

要素のフィルタリング

1.2 ファミリ タイプのリスト – コンポーネント ファミリ

■ ドア タイプの収集

<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>

<VB.NET>

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

</VB.NET>

要素のフィルタリング

2.1 特定オブジェクト クラスのインスタンスのリスト – システム ファミリ

■ 壁インスタンスの収集

<VB.NET>

```
Dim wallCollector = New  
FilteredElementCollector(m_rvtDoc).OfClass(GetType(Wall))  
Dim wallList As IList(Of Element) = wallCollector.ToElements  
</VB.NET>
```

要素のフィルタリング

2.2 特定オブジェクト クラスのインスタンスのリスト – コンポーネント ファミリ

■ ドア インスタンスの収集

<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>

要素のフィルタリング

3.1 特定ファミリタイプの検索 – システム ファミリ タイプ

■ 壁タイプの検索 例) “標準壁:一般 -200mm”

<VB.NET>

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

</VB.NET>

要素のフィルタリング

3.2 特定ファミリタイプの検索 – コンポーネント ファミリ

■ ドア タイプの検索 例) “片側フラッシュ: 0915 x 2134mm”

```
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
    Dim doorTypeList As IList(Of Element) = doorTypeElems.ToList()
    If doorTypeList.Count > 0 Then ' we should have only one.
        doorType1 = doorTypeList(0) ' found it.
    End If
    Return doorType1
End Function
```

要素のフィルタリング

4.1 与えられたファミリタイプのインスタンス検索

- 与えられたタイプでドアを検索 例)“片側フラッシュ: 0915 x 2134mm”

<VB.NET>

```
' Find a list of element with the given Class, family type and Category
(optional).
Function FindInstancesOfType(ByVal targetType As Type, _
    ByVal idFamilyType 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 for accessibility.
    Return elems.ToList()
End Function
```

</VB.NET>

要素のフィルタリング

4.2 与えられたクラスと名前で要素を検索

■ 与えられた名前でレベルを検索

```
<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
    ''' put the result as a list of element for accessibility.
    Return elems.ToList()
End Function
</VB.NET>
```

要素のフィルタリング

その他のオプション

- 使用方法について学習したクラス:
 - FilteredElementCollector
 - ElementClassFilter
 - ElementCategoryFilter
- 異なる多くのフィルタが存在(例):
 - BoundingBoxContainsPointFilter
 - ElementDesignOptionFilter
 - ElementIsCurveDrivenFilter
 - ElementIsElementTypeFilter
 - ElementParameterFilter
 - ...
- 詳細は Revit API 開発者用ガイドの「フィルタリング」を参照

要素の修正



要素の修正方法

要素の修正

要素レベル vs. ドキュメントレベルの修正

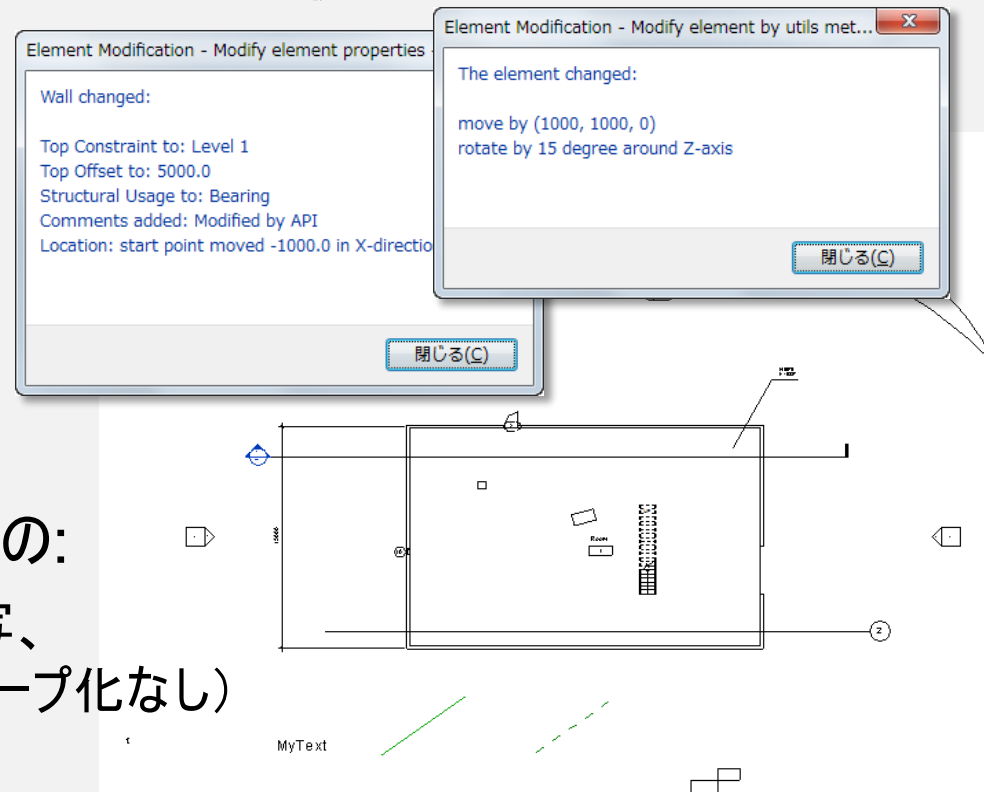
- 要素修正の最初のアプローチ:
 - 各要素レベルでプロパティ、パラメータや位置の変更する
 - 移動や回転など、ドキュメントレベルのメソッドを使用する

- 各要素レベルで可能なもの:

- ファミリタイプ
- パラメータ
- 位置

- ドキュメント メソッドで可能なもの:

- 移動、回転、鏡像化、配列複写、
自動調整なしの配列複写(グループ化なし)



要素の修正

要素レベル – ファミリ タイプ

■ インスタンスのファミリ タイプの変更(例、壁やドア)

<VB.NET>

```
' e.g., an element we are given is a wall.
Dim aWall As Wall = elem
' find a wall family type with the given name.
Dim newWallType As Element = ElementFiltering.FindFamilyType( m_rvtDoc, _
    GetType(WallType), "標準壁", "外壁 - メタル スタッドのレンガ")
' assign a new family type.
aWall.WallType = newWallType
```

</VB.NET>

<VB.NET>

```
' e.g., an element we are given is a door.
Dim aDoor As FamilyInstance = elem
' find a door family type with the given name.
Dim newDoorType As Element = ElementFiltering.FindFamilyType( _
    GetType(FamilySymbol), "片側フラッシュ", "0762 x 2032 mm", _
    BuiltInCategory.OST_Doors)
' assign a new family type.
aDoor.Symbol = newDoorType
```

</VB.NET>

要素の修正

要素レベル - パラメータ

- 要素のパラメータを変更(例、壁やドア)

```
<VB.NET>
```

```
aWall.Parameter(BuiltInParameter.WALL_TOP_OFFSET).Set(14.0)
```

```
aWall.Parameter(BuiltInParameter.ALL_MODEL_INSTANCE_COMMENTS).Set( _  
    "API  で変更")
```

```
</VB.NET>
```

要素の修正

要素レベル – Location Curve

- 位置情報の値を変更(例、壁)

<VB.NET>

```
Dim wallLocation As LocationCurve = aWall.Location
' create a new line bound.
Dim newPt1 = New XYZ(0.0, 0.0, 0.0)
Dim newPt2 = New XYZ(20.0, 0.0, 0.0)
Dim newWallLine As Line = Line.CreateBound(newPt1, newPt2)
' change the curve.
wallLocation.Curve = newWallLine
```

</VB.NET>

要素の修正

ドキュメント レベル – 移動と回転

■ 要素を移動して回転(例、壁)

```
<VB.NET>
    ' move by displacement
    Dim v As XYZ = New XYZ(10.0, 10.0, 0.0)
    m_rvtDoc.Move(elem, v)
</VB.NET>
```

```
<VB.NET>
    ' rotate by 15 degree around z-axis.
    Dim pt1 = XYZ.Zero
    Dim pt2 = XYZ.BasisZ
    Dim axis As Line = Line.CreateBound(pt1, pt2)
    m_rvtDoc.Rotate(elem, axis, Math.PI / 12.0)
</VB.NET>
```


要素の修正

グラフィックスの再作図

- 要素の修正がモデル ジオメトリに反映する際の注意
 - 更新ジオメトリにアクセスする際、グラフィックスを再作図する必要あり
- 再作図は Document.Regenerate() でコントロールが可能
- RegenerationOption.Manual は唯一の再作図オプション ...

```
m_rvtDoc.Regenerate()
```

モデル作成

Revit 要素インスタンスの作成方法

モデル作成

Revit 要素のインスタンス作成

- 新しいジオメトリ要素の作成:

Application.Create.NewXxx() 例、NewBoundingBoxXYZ()

新しいモデル要素の作成:

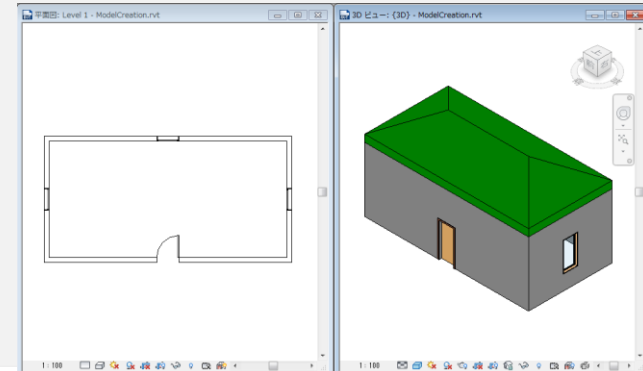
Document.Create.NewXxx() 例、NewFamilyInstance()

スタティックな Create メソッドの使用 例、Wall.Create(doc, ...)

- 特定の要素対応に適用、または
特定の条件に合った複数のオーバーロード メソッド

例、5 Wall.Create()、9 NewFamilyInstance()

詳細は Developer Guide P177 を参照



<VB.NET>

```
' ' create walls
Sub CreateWalls()
    ' get the levels we want to work on.
    Dim level1 As Level = ElementFiltering.FindElement(m_rvtDoc, GetType(Level), "レベル 1")
    Dim level2 As Level = ElementFiltering.FindElement(m_rvtDoc, GetType(Level), "レベル 1")

    ' set four corner of walls.
    Dim pts As New List(Of XYZ) (5)
    ...

    Dim isStructural As Boolean = False ' flag for structural wall or not.

    ' loop through list of points and define four walls.
    For i As Integer = 0 To 3
        ' define a base curve from two points.
        Dim baseCurve As Line = Line.CreateBound(pts(i), pts(i + 1))
        ' create a wall using the one of overloaded methods.
        Dim aWall As Wall = Wall.Create(m_rvtDoc, baseCurve, level1, isStructural)
        ' set the Top Constraint to Level 2
        aWall.Parameter(BuiltInParameter.WALL_HEIGHT_TYPE).Set(level2.Id)
    Next
    ' This is important. we need these lines to have shrinkwrap working.
    m_rvtDoc.Regenerate()
    m_rvtDoc.AutoJoinElements()
End Sub
</VB.NET>
```

<VB.NET>

```
' ' add a door to the center of the given wall.
Sub AddDoor(ByVal hostWall As Wall)

    ' ' get the door type to use.
    Dim doorType As FamilySymbol = _
        ElementFiltering.FindFamilyType(m_rvtDoc, GetType(FamilySymbol), _
            "片側フラッシュ", "0915 x 2134mm", BuiltInCategory.OST_Doors)

    ' ' get the start and end points of the wall.
    Dim locCurve As LocationCurve = hostWall.Location
    Dim pt1 As XYZ = locCurve.Curve.EndPoint(0)
    Dim pt2 As XYZ = locCurve.Curve.EndPoint(1)
    ' ' calculate the mid point.
    Dim pt As XYZ = (pt1 + pt2) / 2.0

    ' ' we want to set the reference as a bottom of the wall or level1.
    Dim idLevel1 As ElementId = _
        hostWall.Parameter(BuiltInParameter.WALL_BASE_CONSTRAINT).AsElementId
    Dim level1 As Level = m_rvtDoc.Element(idLevel1)

    ' ' finally, create a door.
    Dim aDoor As FamilyInstance = m_rvtDoc.Create.NewFamilyInstance( _
        pt, doorType, hostWall, level1, StructuralType.NonStructural)
End Sub
```

</VB.NET>

Revit UI の基礎: リボン、ダイアログ、 選択、イベント



アジェンダ

- UI トピック
 - リボン
 - ユーザ選択
 - タスク ダイアログ
 - イベント
 - ダイナミック モデル アップデート

リボン API

独自のリボン ボタンの追加方法

リボン API

概要

- Ribbon API は唯一の GUI カスタマイズ API
 - メニューとツールバーはリボンに移植する必要あり (Revit 2009 以前の)
- 使用が容易
- WPF 知識の必要はなし
- ガイドラインを用意
 - Autodesk Icon Guidelines.pdf
 - Ribbon design guidelines.pdf

リボン API の概要

- カスタム リボン パネルは既定により [アドイン] タブに追加される
- カスタム リボン パネルは [解析] タブに配置することも可能
- カスタム リボン タブの作成も可能 (Revit 2012 以降、最大 20個)
- 外部コマンドは [アドイン] タブの [外部ツール] 直下に配置される
- 外部アプリケーションはカスタム リボン パネルやタブの利用が可能
 - プッシュ ボタン
 - プルダウン ボタン
 - 単一、または、2～3列のスタック レイアウトが可能
 - スプリット ボタン
 - ラジオ グループ
 - コンボボックス
 - テキストボックス
 - スライド-アウト パネル

リボン API クラス

- RibbonPanel
 - リボンアイテムやボタンを含むパネル
- RibbonItem
 - ボタン、プッシュまたはプルダウン、コンボボックス、テキストボックス、ラジオボタン、その他
- PushButton、PushButtonData
 - プッシュ ボタン情報を管理
- PulldownButton, PulldownButtonData
 - プルダウン ボタン情報を管理
- SplitButton、SplitButtonData
 - スプリット ボタン情報を管理
- ComboBox、ComboBoxData
 - コンボボックス情報を管理
- ...

Revit 2011 以降のリボン API

- 名前空間

Autodesk.Revit.UI

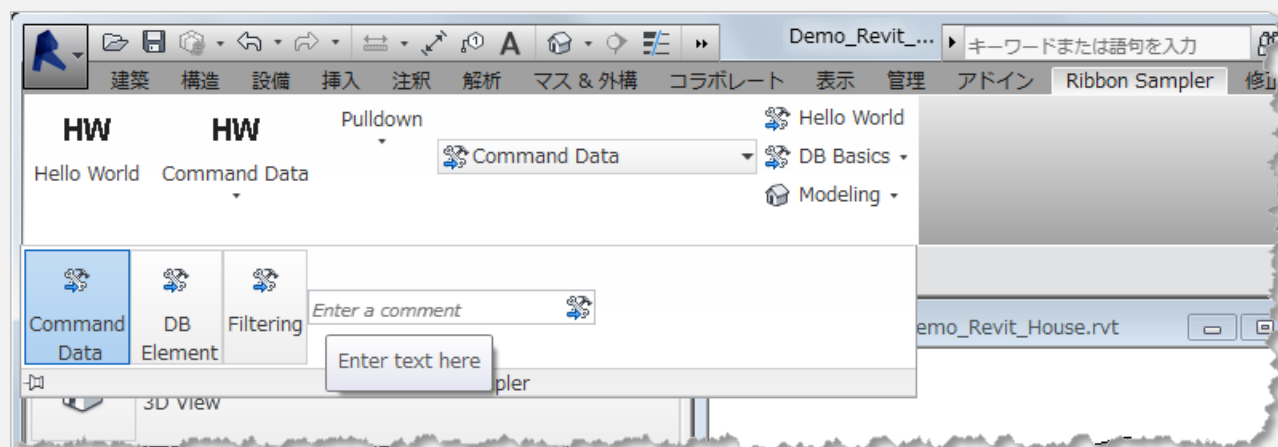
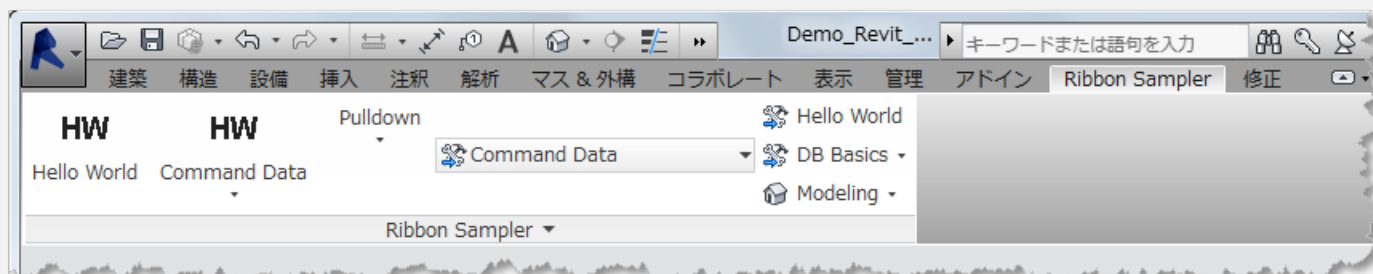
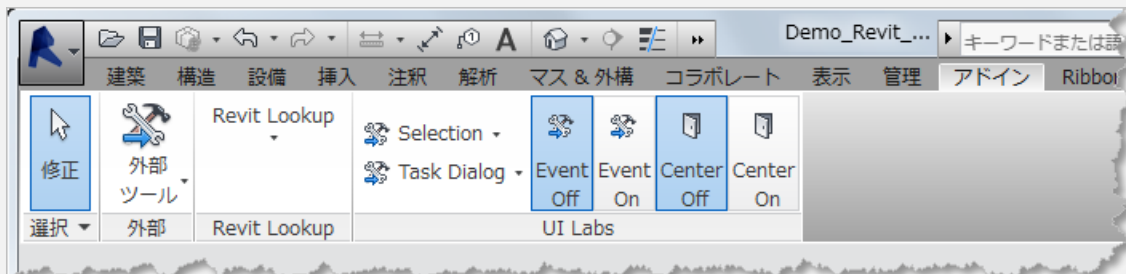
- ウィジェット (SplitButton、ComboBox、TextBox、その他)

- コンボボックスとテキストボックスのイベント

- プロパティ

- RibbonItem.Visible
- RibbonItem.LongDescription
- RibbonItem.ToolTipImage
- PushButton.AvailabilityClassName

実習 – リボン API



ユーザ選択

API を利用した点とオブジェクトの選択

ユーザ選択

概要

- オブジェクト、点、エッジ、面の選択機能を持つ
- アクティブ コレクションに新しい選択を追加:
 - PickObject(), PickObjects()

```
UIDocument uidoc = new UIDocument(document);
Selection choices = uidoc.Selection;

// Choose objects from Revit.

IList<Element> hasPickSome =
    choices.PickElementsByRectangle("Select by rectangle");

if (hasPickSome.Count > 0)
{
    int newSelectionCount = choices.GetElementIds().Count;
    string prompt = string.Format("{0} elements added to Selection.",
        newSelectionCount - selectionCount);
    TaskDialog.Show("Revit", prompt);
}
```

ユーザ選択

概要

- 特定のオブジェクト タイプを選択する機能
 - 要素、要素上の点、エッジ、面
- カスタム ステータス メッセージを追加する機能
 - ステータスバー上の文字
- 選択用のスナップ タイプを定義する機能

```
public void PickPoint(UIDocument uidoc)
{
    ObjectSnapTypes snapTypes =
        ObjectSnapTypes.Endpoints | ObjectSnapTypes.Intersections;

    XYZ point = uidoc.Selection.PickPoint(
        snapTypes, "Select an end point or intersection");

    string strCoords = "Selected point is " + point.ToString();
    TaskDialog.Show("Revit", strCoords);
}
```

- アクティブな作業平面を設定する機能
 - View.SketchPlane()

ユーザ選択

選択フィルタ

- 選択中には ***ISelection*** インタフェースがオブジェクトフィルタの手助け
 - AllowElement()
 - AllowReference()

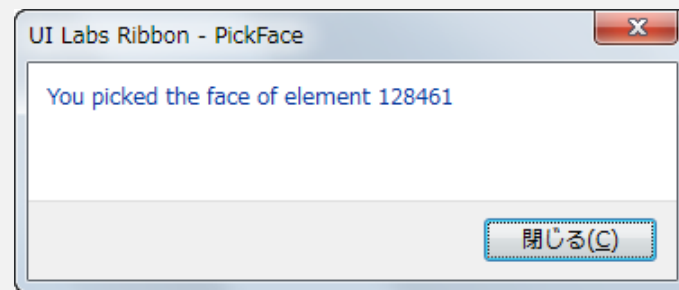
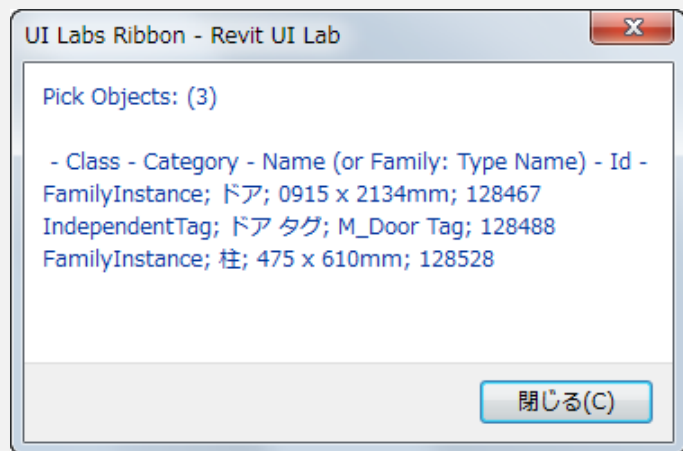
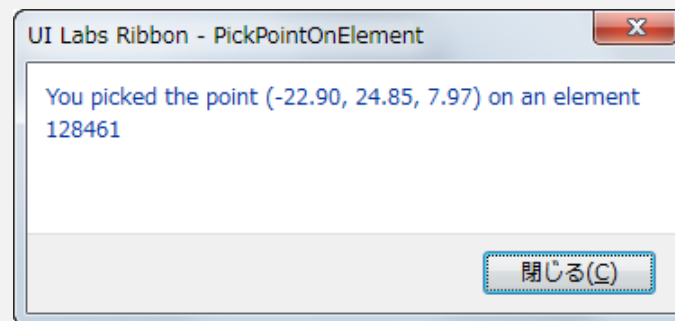
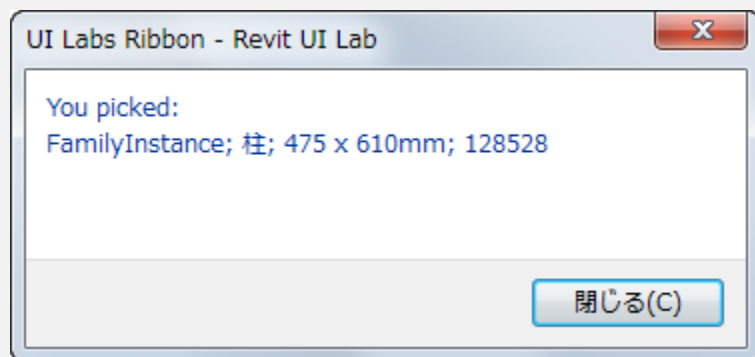
```
public void SelectPlanarFaces(Autodesk.Revit.DB.Document document)
{
    UIDocument uidoc = new UIDocument(document);
    ISelectionFilter selFilter = new PlanarFacesSelectionFilter();
    IList<Reference> faces = uidoc.Selection.PickObjects(
        ObjectType.Face, selFilter, "Select multiple planar faces");
}

public class PlanarFacesSelectionFilter : ISelectionFilter
{
    public bool AllowElement(Element element)
    {
        return true;
    }

    public bool AllowReference(Reference refer, XYZ point)
    {
        if (refer.GeometryObject is PlanarFace) { return true; }
        return false;
    }
}
```

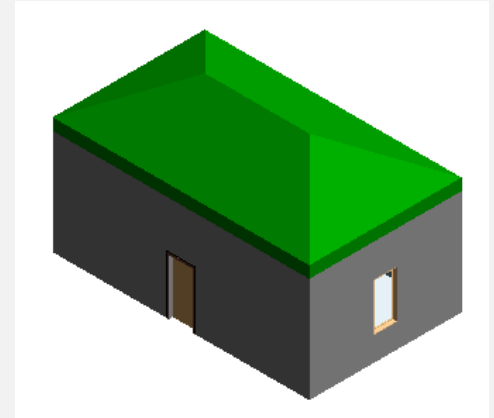
実習 - ユーザ選択

Pick Sampler



実習 – ユーザ選択

House Pick の作成



```
<CS>
XYZ pt1 = rvtUIDoc.Selection.PickPoint("Pick the first corner of walls");
XYZ pt2 = rvtUIDoc.Selection.PickPoint("Pick the second corner");

// simply create four walls with orthogonal rectangular profile
// from the two points picked.
List<Wall> walls = RevitIntroVB.ModelCreation.CreateWalls(
    rvtUIDoc.Document, pt1, pt2);

// pick a wall to add a front door
SelectionFilterWall selFilterWall = new SelectionFilterWall();
Reference @ref = rvtUIDoc.Selection.PickObject(
    ObjectType.Element, selFilterWall, "Select a wall to place a front door");
Wall wallFront = @ref.Element as Wall;

// add a door to the selected wall
RevitIntroVB.ModelCreation.AddDoor(rvtUIDoc.Document, wallFront);
</CS>
```

タスク ダイアログ

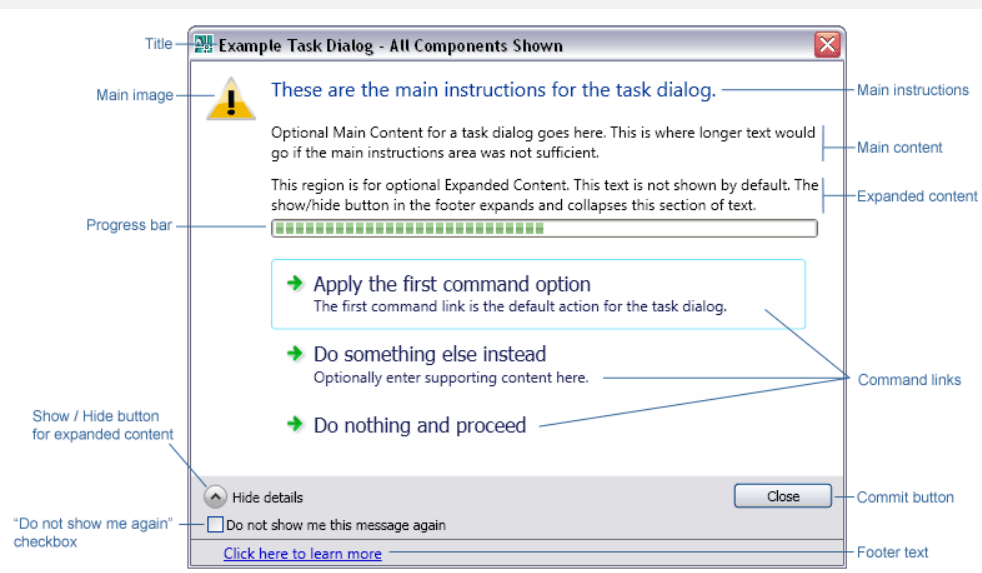


Revit スタイルのメッセージ ボックス

タスク ダイアログ

概要

- コントロール セットを持つモーダル ダイアログ
- 単純な Windows メッセージ ダイアログの代替となる Revit スタイル
- システムが次の項目を必要とする際に使用される ...
 - 情報を提供する
 - 質問する
 - タスク実行のためにユーザにオプションを選択させる



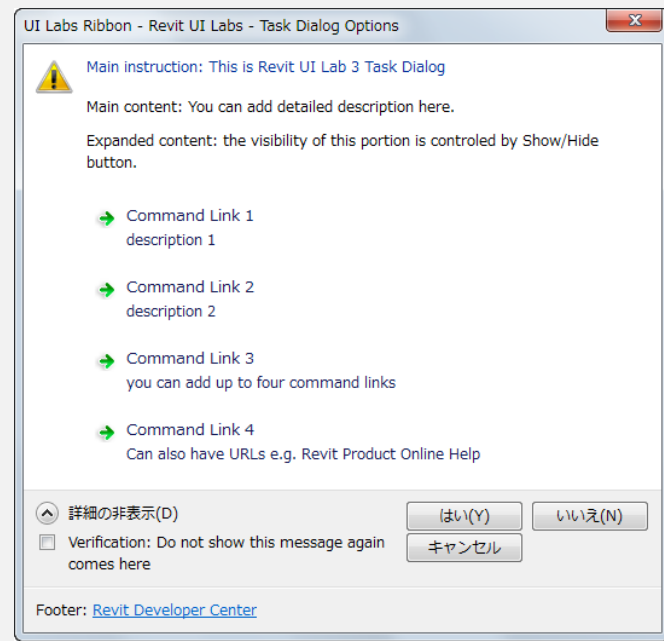
タスク ダイアログ

概要

- タスク ダイアログ作成の2つの方法:
 - TaskDialog を構築してプロパティを指定後に Show() メソッドを呼出
`Autodesk.Revit.UI.TaskDialog` のインスタンス
 - ワンステップで表示するスタティックな Show() メソッドの1つを使用
- 指定可能な情報
 - インストラクション
 - 詳細テキスト
 - アイコン
 - ボタン
 - コマンドリンク
 - 縦書き文字、その他

実習 - タスク ダイアログ

Dialog Sampler



```
<CS>
// (0) create an instance of task dialog to set more options.
TaskDialog myDialog = new TaskDialog("Revit UI Labs - Task Dialog Options");

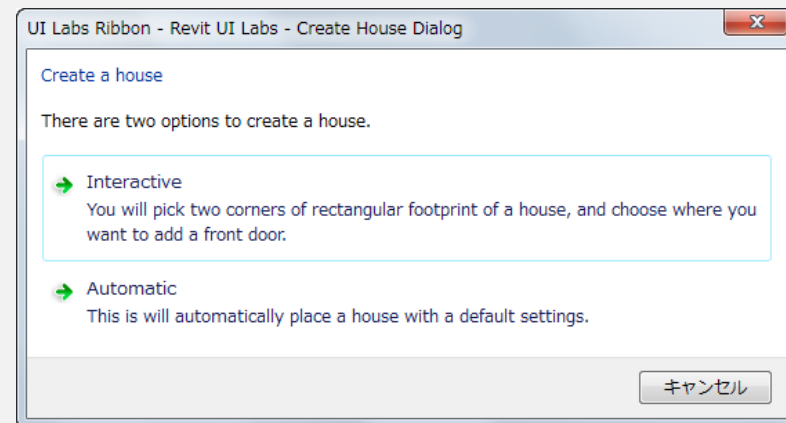
// (1) set the main area. these appear at the upper portion of the dialog.

myDialog.MainIcon = TaskDialogIcon.TaskDialogIconWarning;
// or TaskDialogIcon.TaskDialogIconNone.
myDialog.MainInstruction =
    "Main instruction: This is Revit UI Lab 3 Task Dialog";
myDialog.MainContent = "Main content: You can add detailed description here.";

if (stepByStep) myDialog.Show();
</CS>
```

実習 - タスク ダイアログ

Create House Dialog



```
<CS>
    TaskDialog houseDialog = new TaskDialog("Revit UI Labs - Create House
Dialog");
    houseDialog.MainInstruction = "Create a house";
    houseDialog.MainContent = "There are two options to create a house.";
    houseDialog.AddCommandLink(TaskDialogCommandLinkId.CommandLink1,
    "Interactive", "You will pick two corners of rectangular footprint of a house, and
choose where you want to add a front door.");
    houseDialog.AddCommandLink(TaskDialogCommandLinkId.CommandLink2, "Automatic",
    "This is will automatically place a house with a default settings.");
    houseDialog.CommonButtons = TaskDialogCommonButtons.Cancel;
    houseDialog.DefaultButton = TaskDialogResult.CommandLink1;

    // show the dialog to the user.
    TaskDialogResult res = houseDialog.Show();
</CS>
```


イベントとダイナミック モデル アップデート

アプリケーション、ドキュメント、要素イベント

イベント

概要

- 特定のアクションで発生する通知
- .NET イベント標準に準拠
 - プレとポスト イベント
 - 単一イベント (DocumentChanged と FailureProcessing)
- タイプ :
 - アプリケーション レベル
 - ドキュメント レベル
 - 要素 レベル

イベント

概要

- DB イベントと UI イベントの分類
 - DB イベント Application クラスと Document クラスから利用可能
 - UI イベント UIApplication クラスから利用可能
- イベント中のモデル編集確認
 - Document.IsModifiable
 - Document.IsReadOnly
- 新しいプレ イベントの多くはキャンセルが可能
 - RevitEventArgs.Cancellable
 - RevitAPIPreEventArgs.Cancel

イベント

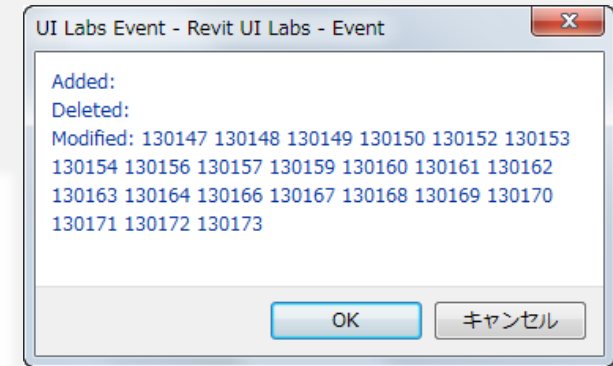
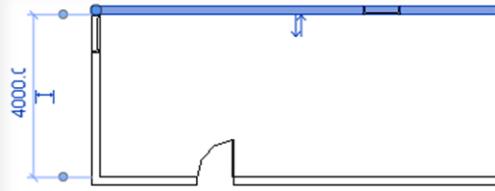
イベント ハンドラ、イベントの登録と登録解除

```
public void UILabs_DocumentChanged(object sender, DocumentChangedEventArgs args)
{
    // Do something here
}
```

```
public Result OnStartup(UIControlledApplication application)
{
    application.ControlledApplication.DocumentChanged += UILabs_DocumentChanged;
    return Result.Succeeded;
}
```

```
public Result OnShutdown(UIControlledApplication application)
{
    application.ControlledApplication.DocumentChanged -= UILabs_DocumentChanged;
    return Result.Succeeded;
}
```

実習 - イベント



```
// register the document changed event  
application.ControlledApplication.DocumentChanged += UI Labs _DocumentChanged;
```

```
// you can get the list of ids of element added/changed/modified.  
Document rvtdDoc = args.GetDocument();  
  
ICollection<ElementId> idsAdded = args.GetAddedElementIds();  
ICollection<ElementId> idsDeleted = args.GetDeletedElementIds();  
ICollection<ElementId> idsModified = args.GetModifiedElementIds();  
  
// put it in a string to show to the user.  
string msg = "Added: ";  
foreach (ElementId id in idsAdded)  
{  
    msg += id.IntegerValue.ToString() + " ";  
}
```

ダイナミック モデル アップデートの概要

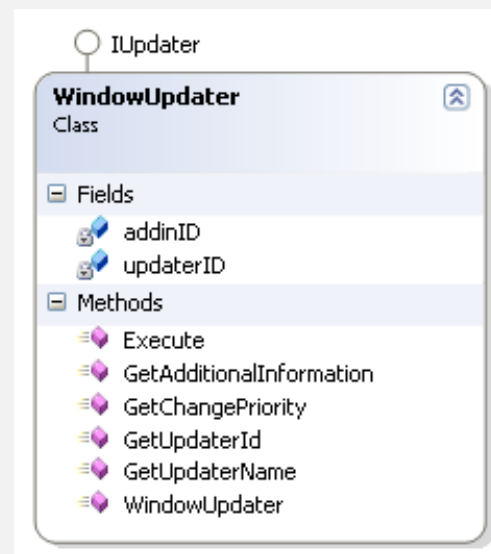
- “モデル内で発生する変更に対応する、Revit API アプリケーション
Revit モデル修正機能”
- 要素の追加、修正、削除を追跡する助けとなります

ダイナミック モデル アップデート

アップデーター

アップデーター :

- 変更のスコープ(範囲)を通知するメソッドを実装する機能
- *IUpdater* インタフェースの実装
 - GetUpdaterId()
 - GetUpdaterName()
 - GetAdditionalInformation()
 - GetChangePriority()
 - Execute()



ダイナミック モデル アップデート

登録とトリガ

- アップデーターの登録

- OnStartup はアプリケーション レベルのスコープ
- ExternalCommand はコマンド レベルのスコープ

```
WindowUpdater updater = new WindowUpdater(application.ActiveAddInId );  
// Register the updater in the singleton UpdateRegistry class  
UpdaterRegistry.RegisterUpdater( updater );
```

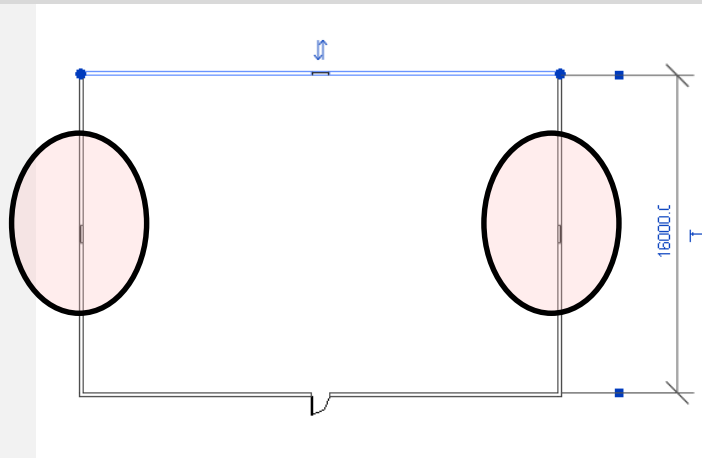
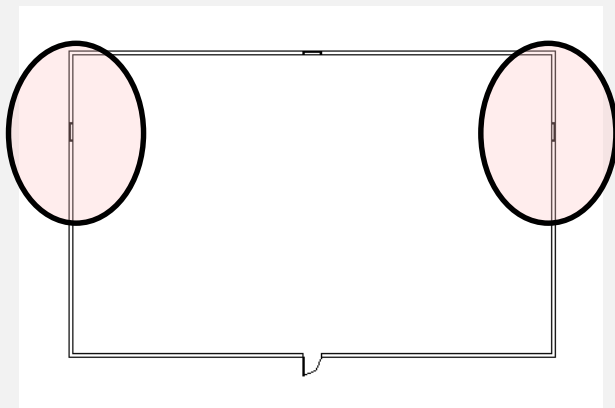
- トリガの追加

- スコープの変更 – ElementId リスト、または ElementFilter を介した要素のリスト
- 変更タイプ – 追加、削除、修正

```
// Set the filter  
ElementClassFilter filter = new ElementClassFilter( typeof( Wall ) );  
// Add trigger  
UpdaterRegistry.AddTrigger(updater.GetUpdaterId(),filter,  
Element.GetChangeTypeGeometry());
```


実習 - ダイナミック モデル アップデート

```
// construct our updater.  
WindowDoorUpdater winDoorUpdater =  
    new WindowDoorUpdater(application.ActiveAddInId);  
  
// ActiveAddInId is from addin manifest. register it  
UpdaterRegistry.RegisterUpdater(winDoorUpdater);  
  
// tell which elements we are interested in notified.  
// we want to know when wall changes it's length.  
  
ElementClassFilter wallFilter = new ElementClassFilter(typeof(Wall));  
UpdaterRegistry.AddTrigger(  
    winDoorUpdater.GetUpdaterId(), wallFilter,  
    Element.GetChangeTypeGeometry());
```



結論

次に目指す場所は ...

既にカバーした内容...

- UI トピック
 - リボン
 - ユーザ選択
 - タスク ダイアログ
 - イベント
 - ダイナミック モデル アップデート

参考 Web ページ

- Revit API 開発者用ガイド(20xx を西暦年に置換してください)
 - <http://help.autodesk.com/view/RVT/20xx/JPN/?guid=GUID-F0A122E0-E556-4D0D-9D0F-7E72A9315A42>
- Revit 開発関連ブログ
 - Technology Perspective from Japan
 - http://adndevblog.typepad.com/technology_perspective/
 - The Building Coder, Jeremy Tammik's Revit API
 - <http://thebuildingcoder.typepad.com>
- ディスカッション グループ
 - <http://discussion.autodesk.com> > Revit Architecture > Revit API
- .NET 言語変換
 - Convert C# to VB.NET
 - <http://www.developerfusion.com/tools/convert/csharp-to-vb/>
 - Convert VB.NET to C#
 - <http://www.developerfusion.com/tools/convert/vb-to-csharp/>

ファミリ API



Revit ファミリ API

■ 背景

- ファミリ コンテンツ作成は API なしでも高いカスタマイズ機能を保持
- UI 上での動作の理解は API を使ったファミリ作成時のキーとなり得る
- Revit API の知識の2つのコミュニティが存在
 - UI とコンテンツ作成をよく理解している
 - プログラミングが得意だが UI に精通していない

■ ゴール

- UI を利用したベスト プラクティスに沿ったファミリ API の学習
- ファミリ API を使ったプログラム作成に従った手順の確立
 - 安定性と柔軟性、適応性を持たせることが重要

アジェンダ

- UI からの Revit ファミリ
 - ファミリとは？
 - どこから初めて、振る舞い、エディタ、何が可能なのか
 - ベスト プラクティス
- API を使ったファミリー作成
 - ベスト プラクティスに沿った習得
 - 例: L-形柱
 - 学習リソース

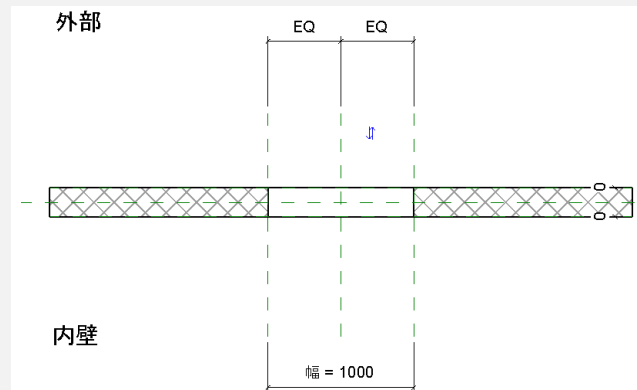
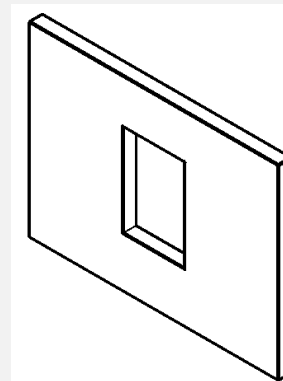
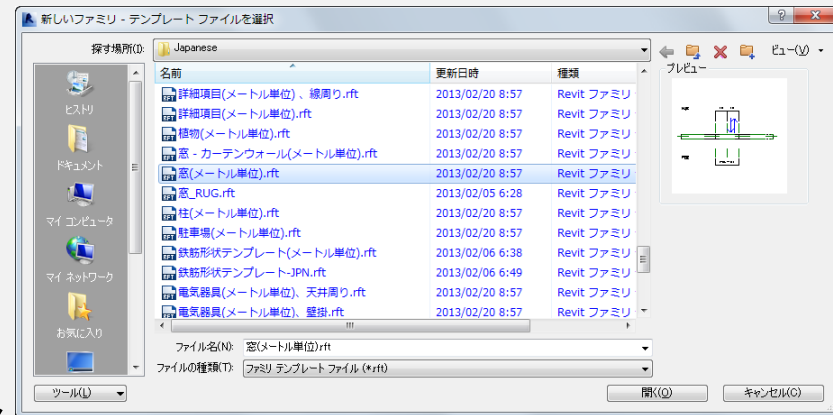
Revit ファミリ – ファミリとは？

- 建設オブジェクトやシンボルのグラフィック表現
 - 3D または 2D ジオメトリ
 - オブジェクトの定義/作成をサポートするデータ
- 一般的に - 3 タイプ
 - システム ファミリ – プロジェクト テンプレート内に保存されている
 - 壁、屋根、床、天井 ...
 - インプレイス ファミリ – “オブジェクトの一種”
 - 標準ファミリ – “.rfa” ファイルとして独立している
 - 窓、ドア、家具、梁、照明器具 ...
 - Revit 2010 から API を提供



Revit ファミリ – どこから始める ?

- どちらがベターか ?
 - ファミリ テンプレートから始める
 - 既存ファミリの修正
- どのファミリ テンプレートで始めるか ?
 - 2D や 3D、モデルや詳細コンポーネント
 - ホストされるものか、されないものか:
 - 壁、天井、面ベース ...
 - カテゴリ
 - 配置タイプ: 自由、または、2点
 - 特殊性: トラス、鉄筋 ...



Revit ファミリの振る舞い

■ Revit Architecture

- モデル内で単純化された標準ビルディング コンポーネント
- 自由な配置が可能なオブジェクト - 据付棚、家具、その他
- “2 点” 配置のオブジェクト - 梁、詳細コンポーネント、その他
- ホストされるオブジェクト: 窓、ドア、柱 (“レベルからレベル”)、天井や壁ベースの照明器具

■ Revit Structure

- 他のオブジェクトと複雑に相互作用する追加コンポーネント
- フレーム - 梁 (“梁から梁”、“梁から柱”)、柱
- トラス - 大梁トラスのレイアウト; 境界条件
- スパン方向記号; 鉄筋の記号 - 面の配筋はエッジ検索用に拡張、パス配筋

■ Revit MEP

- 何に接続するかによってコネクタはオブジェクトのサイズ変更を許容

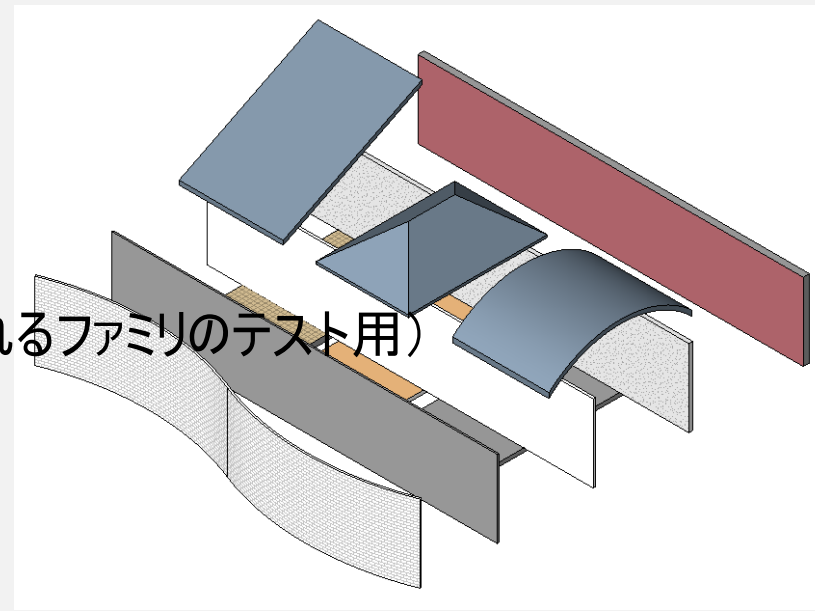
Revit ファミリ エディタ

- Revit は 6 種類ファミリ エディタを提供
 - 3D モデル、注釈、詳細、鉄筋、トラス、コンセプト マス
- 各ファミリ エディタが提供するもの
 - 特定のセットと選択されたファミリ テンプレートへの関連性
 - ジオメトリ – 押し出し、ブレンド、スイープ、回転、スイープ ブレンド
 - 線分 – モデル、シンボリック、詳細
 - 標準ツール – コピー、鏡像、ペイント、結合/分離、切り取り/切り取り解除
 - 参照 – 参照面、参照線
 - 注釈ツール – ラベル
 - 高度なツール – 計算式、ネスト、配列、タイプ カテゴリ
 - MEP ツール – コネクタの追加

Revit ファミリのベスト プラクティス

ビルディング ファミリの“処理”は習得が必要なファミリ作成の最重要概念
処理手順:

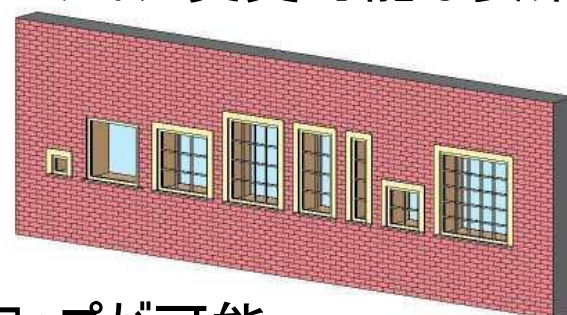
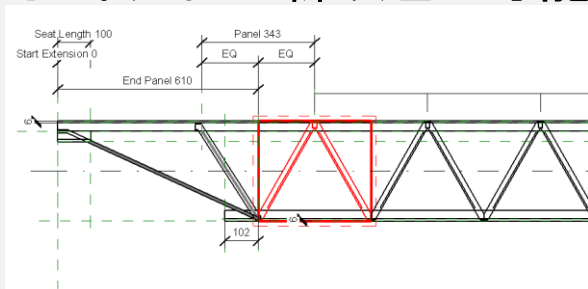
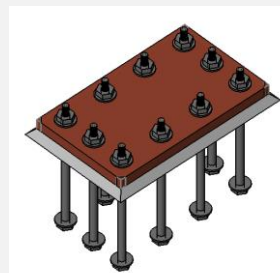
1. 計画(挿入点、パラメトリック原点)
2. レイアウト参照面(縦骨など)
3. パラメータの追加
4. 複数のホスト厚タイプの追加(ホストされるファミリのテスト用)
5. 2つ以上のタイプの追加
6. フレキシブル タイプとホスト(テスト手順)
7. 単一レベルのジオメトリの追加
8. 満足が得られるまで 6 と 7 の繰り返し
9. プロジェクト環境でのテスト(テスト プロジェクトの作成)



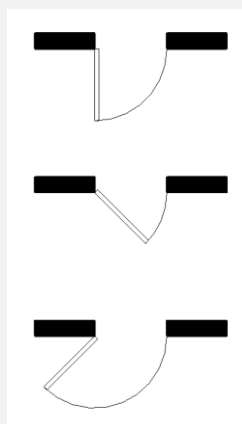
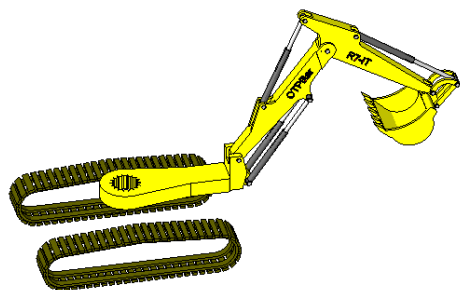
Steven Campbell, Revit Content Project Manager

Revit ファミリー – なにが可能か

- 計算式 – 振る舞い、表示、配列化の制御が可能
- 配列とネスト – 繰り返し可能、配列間のサイズ変更可能な要素

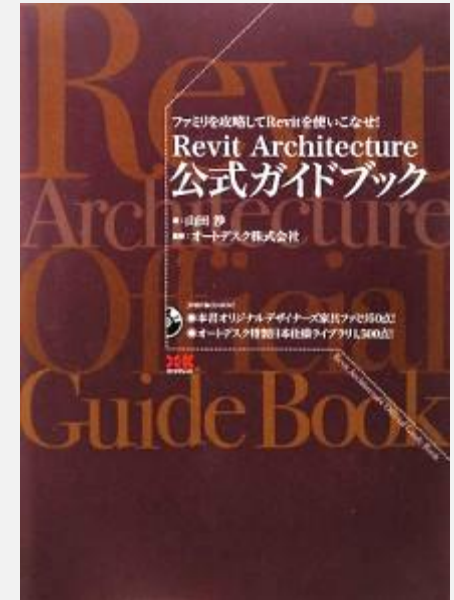


- 高度なネスト – サブコンポーネントのスワップが可能
- 参照線 – 角度方向の移動



ファミリ リソース

- Revit ファミリ ガイド
 - 2009 & 2010 バージョン
- ニュースグループ
 - discussion.autodesk.com
 - AUGI - <http://forums.augi.com>
- 市販本 & DVD
 - Revit Architecture 公式ガイドブック – 山田 渉
 - Mastering Autodesk Revit Building – Paul F. Aubin
 - Mastering Family Editor Series - 5 DVD's – Paul F. Aubin
 - [Learning Autodesk Revit MEP 2012](http://cad-notes.com/2011/12/learning-autodesk-revit-mep-2012-training-video-is-available) video training by Simon Whitbread, Don Bokmiller and Joel Londenbergl
<http://cad-notes.com/2011/12/learning-autodesk-revit-mep-2012-training-video-is-available>



ファミリ API



ベスト プラクティスに沿った基礎を習得

ファミリ API の概要

どのようなものか？

- ファミリ エディタ コンテキスト内の Revit API
- ライブラリ生成の自動化:
 - ファミリ ライブラリを自動的に生成
 - 場合によって他のライブラリ仕様とリンク
- 情報の展開と修正
- UI に類似した操作
- マニュアル操作との差異や制限が存在

ファミリ API の概要

ファミリ仕様クラスとメソッド

- FamilyManager クラス:
 - タイプの追加/削除/名前変更
 - パラメータの追加/削除
 - 値と計算式の設定
- ファミリ コンテキスト固有の Document メソッド:
 - IsFamilyDocument
 - 現在のドキュメントがファミリ ドキュメントかを識別
 - OwnerFamily
 - このファミリ ドキュメントを所有するファミリを返す
 - FamilyManager
 - ファミリ タイプとパラメータへのアクセスを提供する FamilyManager オブジェクトを返す
 - FamilyCreate
 - ファミリ ドキュメント内に新しい要素インスタンスを作成するために FamilyItemCreate オブジェクトを返す
 - プロジェクトの Create オブジェクトに相当
 - EditFamily
 - プロジェクト ドキュメントにロードされたファミリを編集

実習

ハンズオン

ファミリ API 実習

ハンズオン

- 柱ファミリ作成による実習:
 - Lab1 – 矩形プロファイルで柱ファミリを作成
 - Lab2 – L-形プロファイルで柱ファミリを作成
 - Lab3 – 計算式の追加とマテリアルの割り当て
 - Lab4 – 可視性コントロールの追加
- ステップ毎の手順を含む
- VB.NET と C#

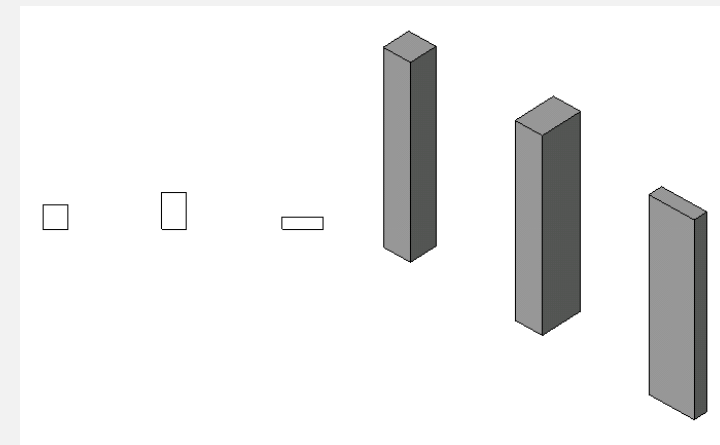
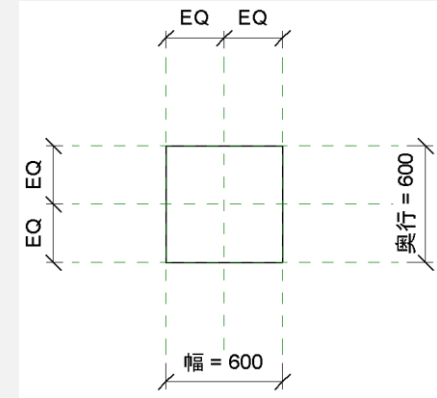
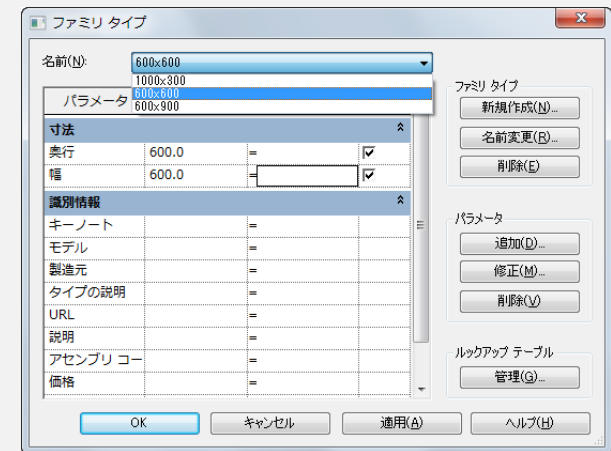
Lab1 – 長方形柱の作成

目的: ファミリ API の基礎を習得

- ファミリ コンテキストのチェック
- 押し出しを使った単純なソリッドの作成
- 位置合わせの設定
- タイプの追加

クラスとメソッド:

- `rvtDoc.IsFamilyDocument()`
- `rvtDoc.OwnerFamily.FamilyCategory.Name`
- `rvtDoc.FamilyCreate.NewExtrusion()`
- `rvtDoc.FamilyCreate.NewAlignment()`
- `familyMgr = rvtDoc.FamilyManager`
- `familyMgr.NewType()`
- `familyMgr.Parameter(); familyMgr.Set()`



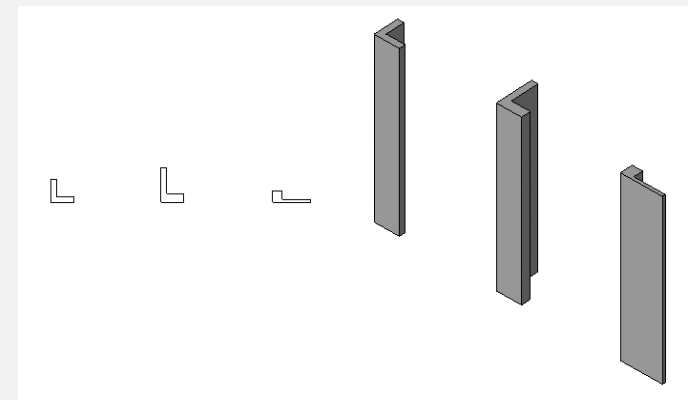
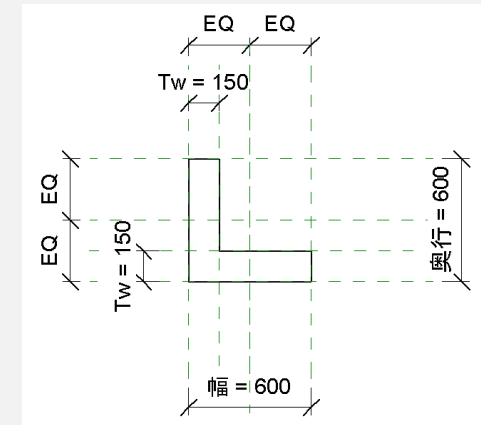
Lab2 – L-形柱の作成

目的:ファミリ API の基礎を習得

- 参照面の追加
- パラメータの追加
- 寸法の追加

クラスとメソッド:

- `rvtDoc.FamilyCreate.NewReferencePlane()`
- `familyMgr.AddParameter()`
- `rvtDoc.FamilyCreate.NewDimension()`



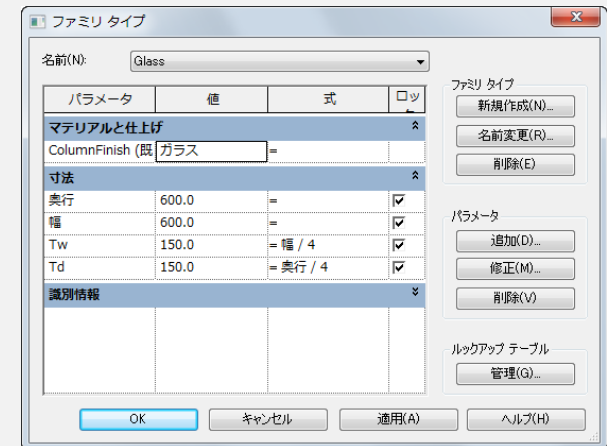
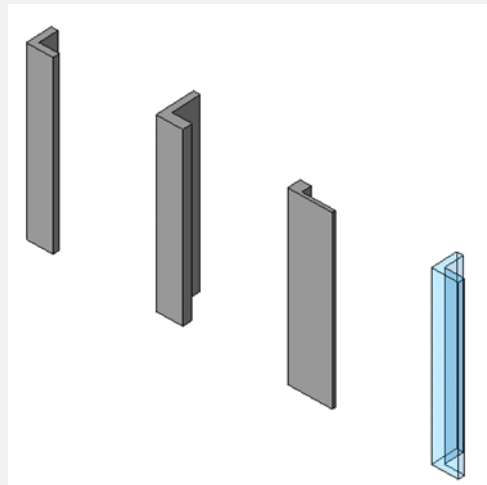
Lab3 – 計算式とマテリアルの追加

目的:ファミリ API の基礎を習得

- 計算式の追加
- マテリアルの追加

クラスとメソッド:

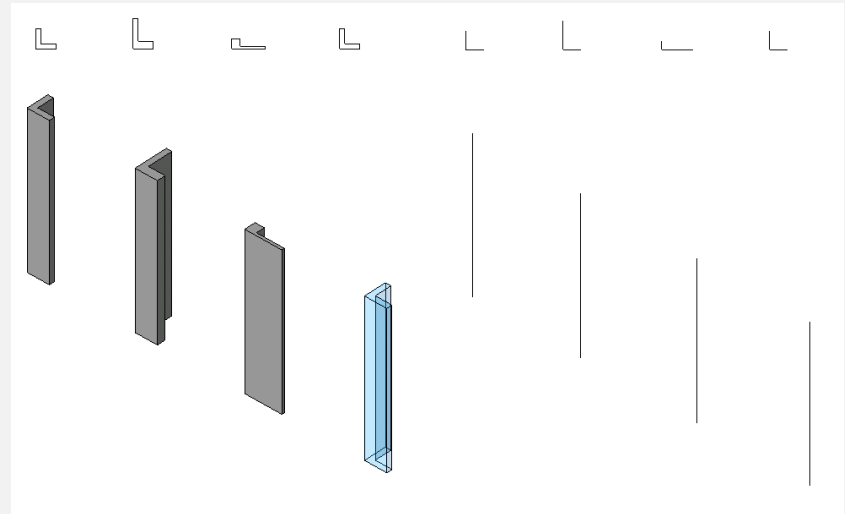
- `familyMgr.SetFormula()`
- `pSolid.Parameter("Material")`
- `familyMgr.AddParameter("MyColumnFinish", BuiltInParameterGroup.PG_MATERIALS, ParameterType.Material, True)`
- `familyMgr.AssociateElementParameterToFamilyParameter()`



Lab4 – 可視性コントロールの追加

目的:

- 線表現の追加
- 可視性コントロールの追加

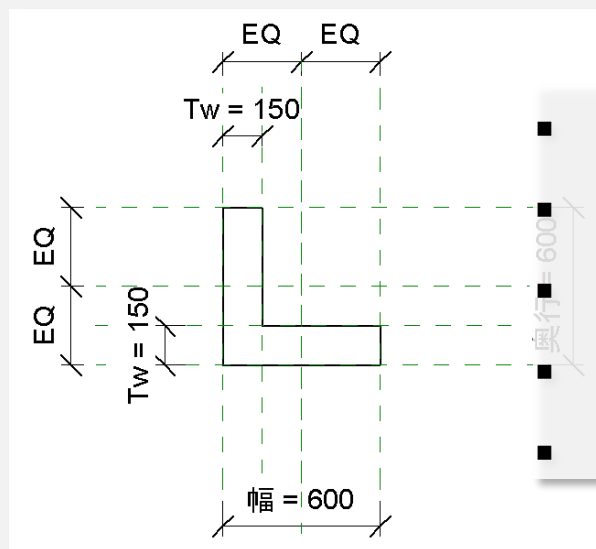


クラスとメソッド:

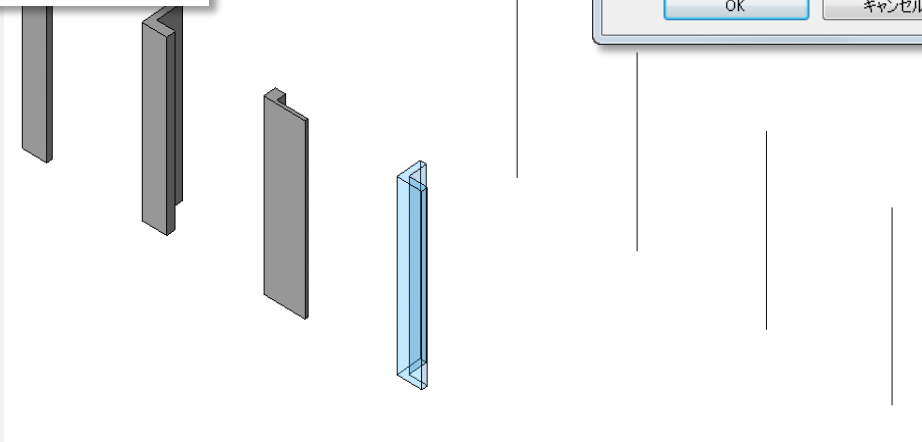
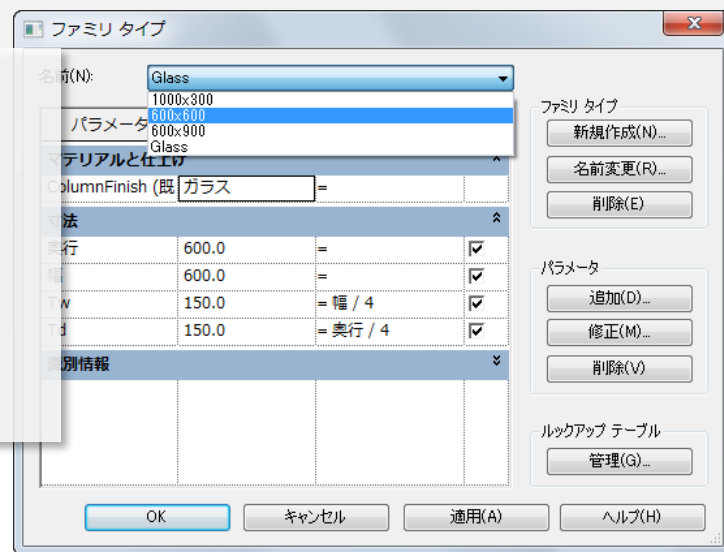
- `rvtDoc.FamilyCreate.NewSymbolicCurve()`
- `rvtDoc.FamilyCreate.NewModelCurve()`
- `FamilyElementVisibility(FamilyElementVisibilityType.ViewSpecific/Model)`
- `FamilyElementVisibility.IsShownInFine`, etc.
- `pLine.SetVisibility(pFamilyElementVisibility)`

ベスト プラクティスに沿ったファミリ API

例: 単純な L-形柱

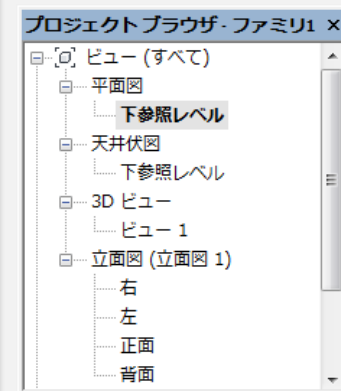
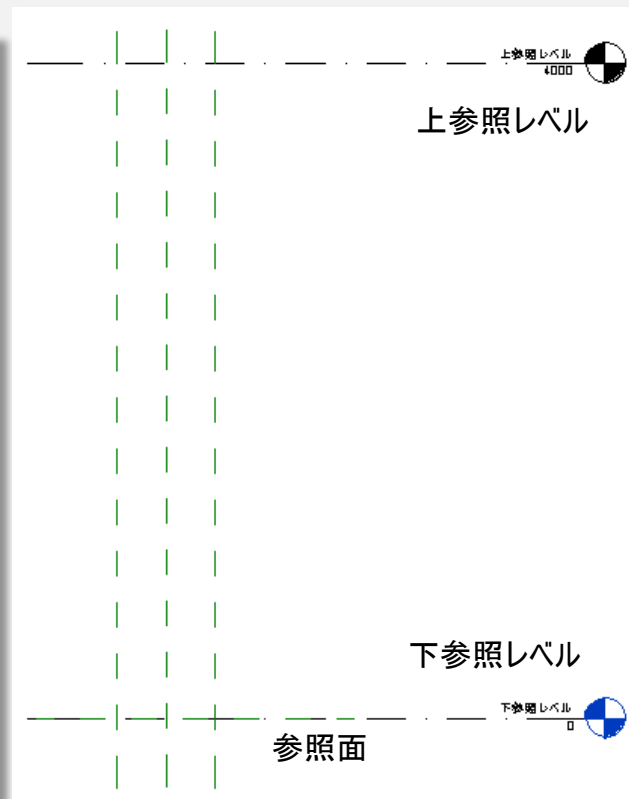
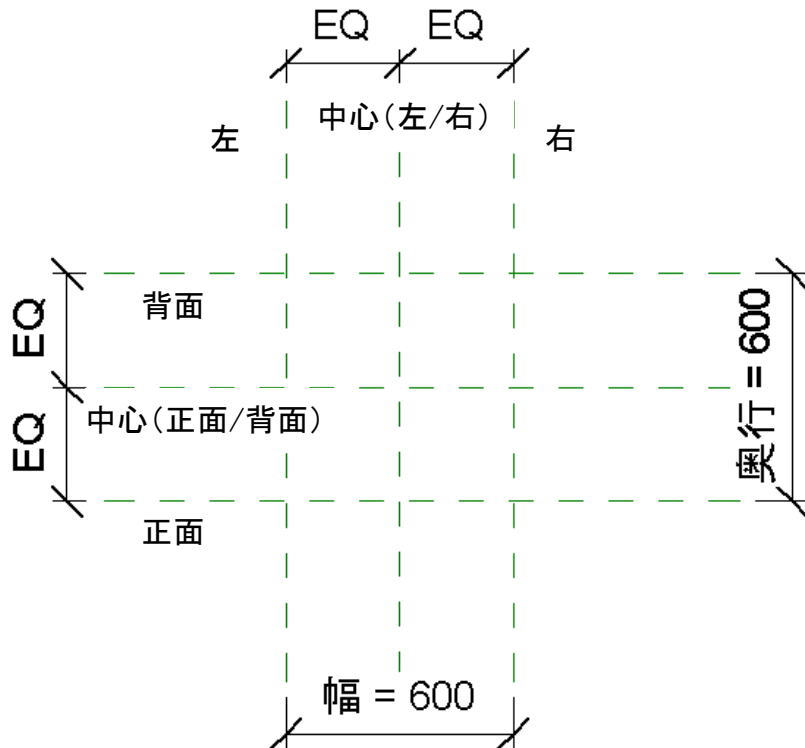


- 計画
- 参照面
- パラメータ
- タイプ
- ジオメトリ



1.計画

- 正しいテンプレート(例、“柱(メートル単位).rft”)
- 既存テンプレートの理解



1.計画

ドキュメントの検証

```
Function ValidateDocument(ByVal rvtDoc As Document) As Boolean
    ' our command works in the context of family editor only
    If Not rvtDoc.IsFamilyDocument Then
        TaskDialog.Show("Family API", "This works only in the family editor.")
        Return False
    End If
    ' check if we have a right template
    Dim ownerFamily As Family = rvtDoc.OwnerFamily
    If ownerFamily Is Nothing Then
        TaskDialog.Show("Family API", "This document does not have Owner Family.")
        Return False
    End If
    ' check the family category of this document
    Dim catColumn As Category = _
        rvtDoc.Settings.Categories.Item(BuiltInCategory.OST_Columns)
    If Not ownerFamily.FamilyCategory.Id.Equals(catColumn.Id) Then
        TaskDialog.Show("Family API", "Please open 柱(メートル単位).rft")
    End If
    Return True
End Function
```

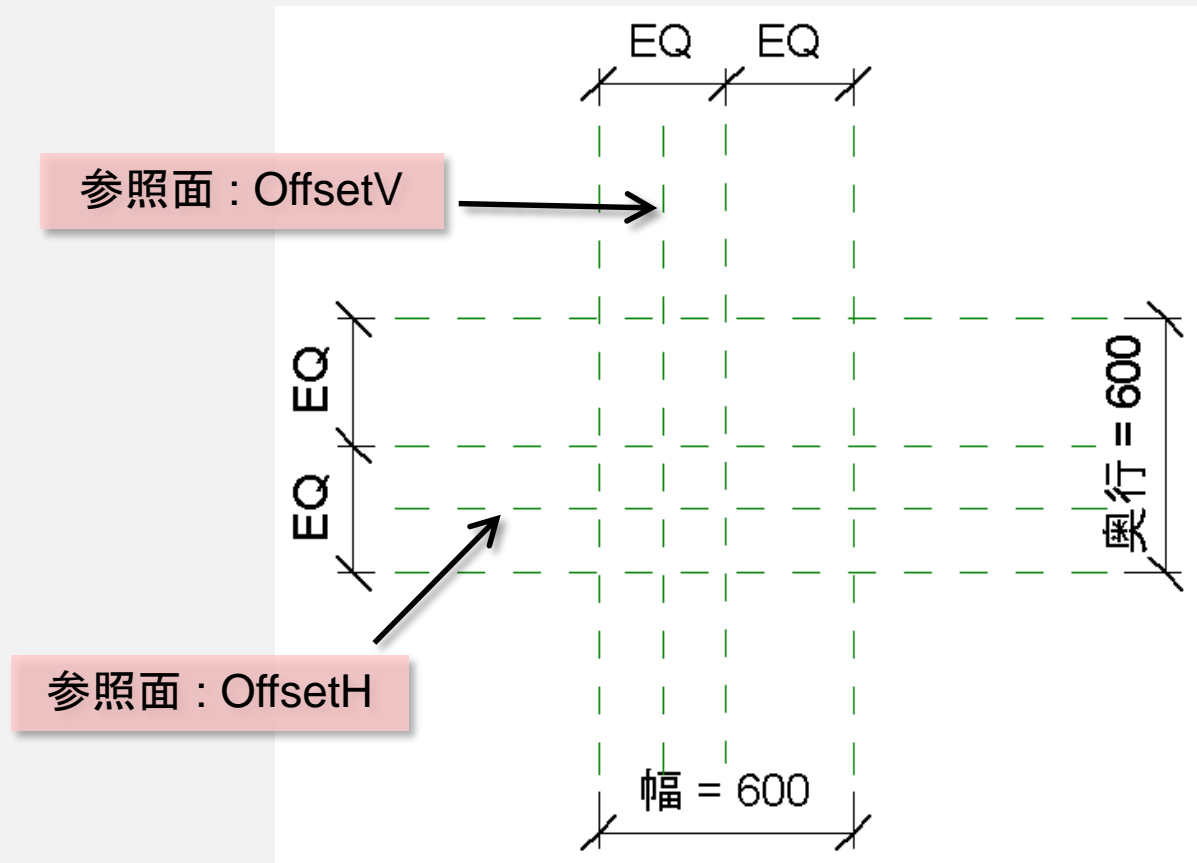
ドキュメントがファミリ ドキュメントかをチェック

このドキュメントのファミリを取得

このドキュメントのファミリ カテゴリをチェック

2.参照面のレイアウト

参照面の追加



2.参照面のレイアウト

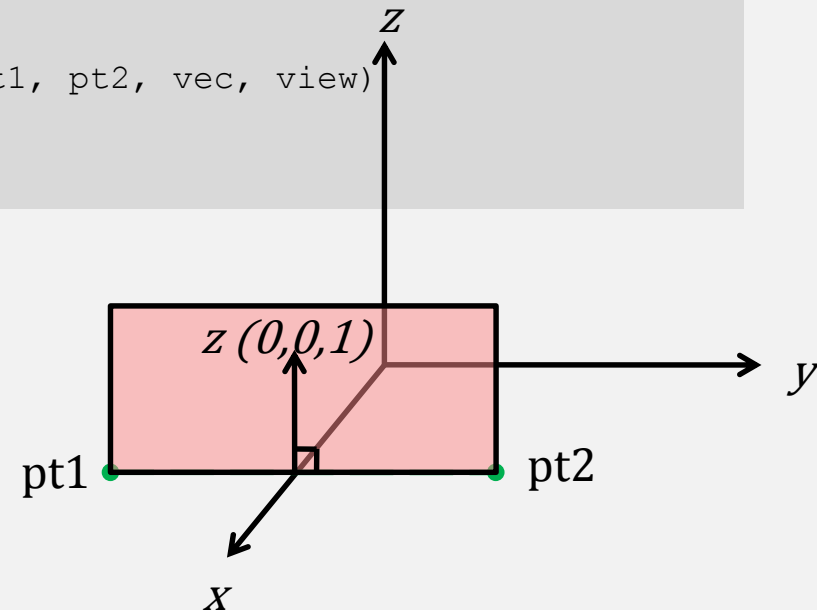
例: 縦方向のオフセット

```
Sub AddReferencePlane_VerticalOffset()  
    '' create a reference plan, using NewReferencePlane  
    Dim pt1 As New XYZ(-0.5, -2.0, 0.0) '' one end  
    Dim pt2 As New XYZ(-0.5, 2.0, 0.0) '' the other end  
    Dim vec As XYZ = XYZ.BasisZ          '' perpendicular to the first line.  
    Dim view As View = _  
        Utils.FindElement(rvtDoc, GetType(ViewPlan), "下参照レベル")  
  
    Dim refPlane As ReferencePlane = _  
        m_rvtDoc.FamilyCreate.NewReferencePlane(pt1, pt2, vec, view)  
    refPlane.Name = "OffsetV"  
End Sub
```

2.参照面のレイアウト

例: 縦方向のオフセット

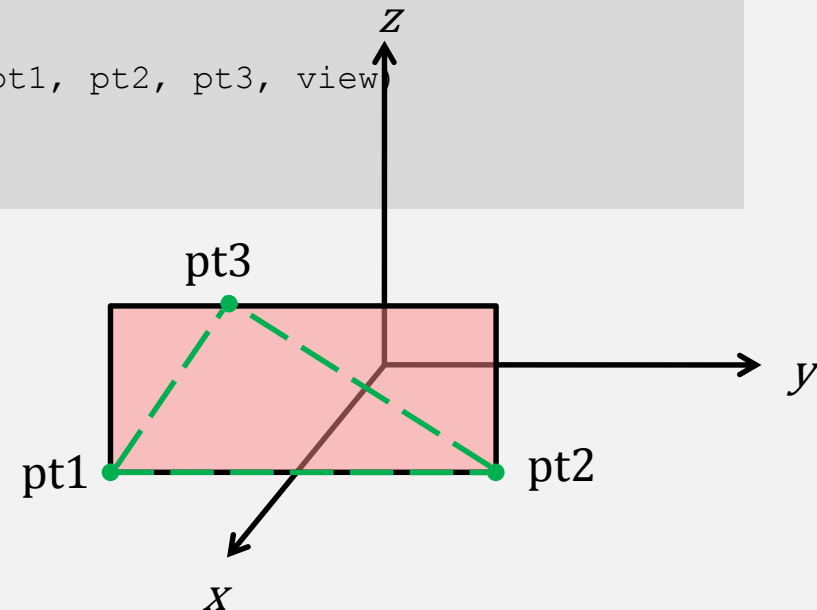
```
Sub AddReferencePlane_VerticalOffset()  
    '' create a reference plan, using NewReferencePlane  
    Dim pt1 As New XYZ(-0.5, -2.0, 0.0) '' one end  
    Dim pt2 As New XYZ(-0.5, 2.0, 0.0) '' the other end  
    Dim vec As XYZ = XYZ.BasisZ          '' perpendicular to the first line.  
    Dim view As View = _  
        Utils.FindElement(rvtDoc, GetType(ViewPlan), "下参照レベル")  
  
    Dim refPlane As ReferencePlane = _  
        m_rvtDoc.FamilyCreate.NewReferencePlane(pt1, pt2, vec, view)  
    refPlane.Name = "OffsetV"  
End Sub
```



2.参照面のレイアウト

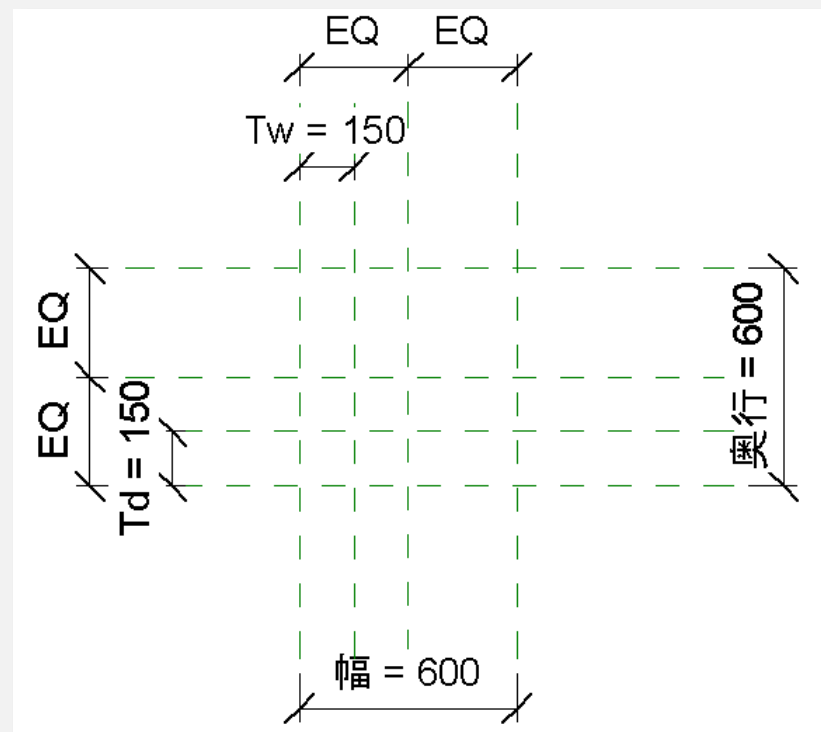
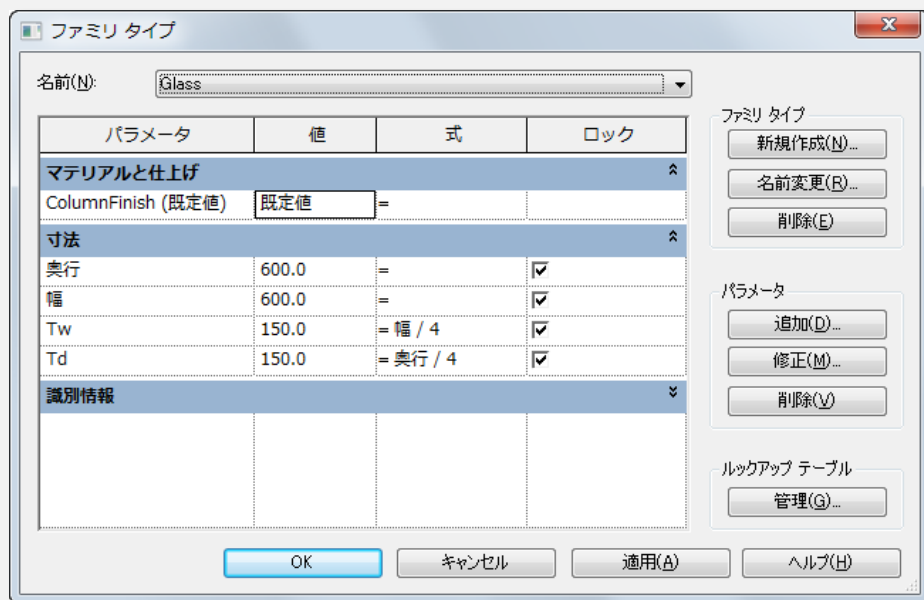
例: NewReferencePlane2()

```
Sub AddReferencePlane_VerticalOffset2()  
    '' create a reference plan, using NewReferencePlane  
    Dim pt1 As New XYZ(-0.5, -2.0, 0.0) '' one end  
    Dim pt2 As New XYZ(-0.5, 2.0, 0.0) '' the other end  
    Dim pt3 As New XYZ(-0.5, -1.0, 1.0) '' the third point  
    Dim view As View = _  
        Utils.FindElement(rvtDoc, GetType(ViewPlan), "下参照レベル")  
  
    Dim refPlane As ReferencePlane = _  
        m_rvtDoc.FamilyCreate.NewReferencePlane2(pt1, pt2, pt3, view)  
    refPlane.Name = "OffsetV"  
End Sub
```



3.パラメータの追加

- パラメータ
- 寸法

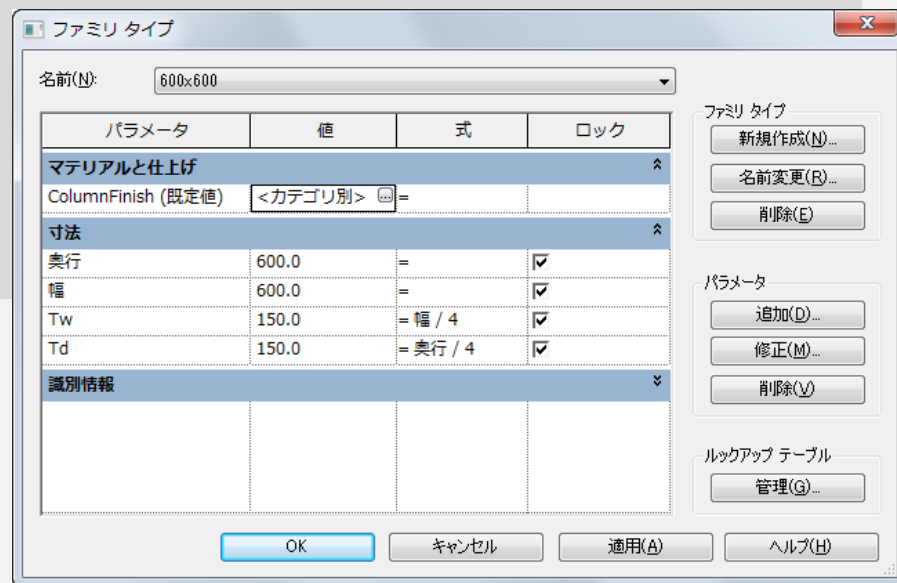


3a.パラメータの追加

例: Tw

```
Dim m_familyMgr As FamilyManager = m_rvtDoc.FamilyManager

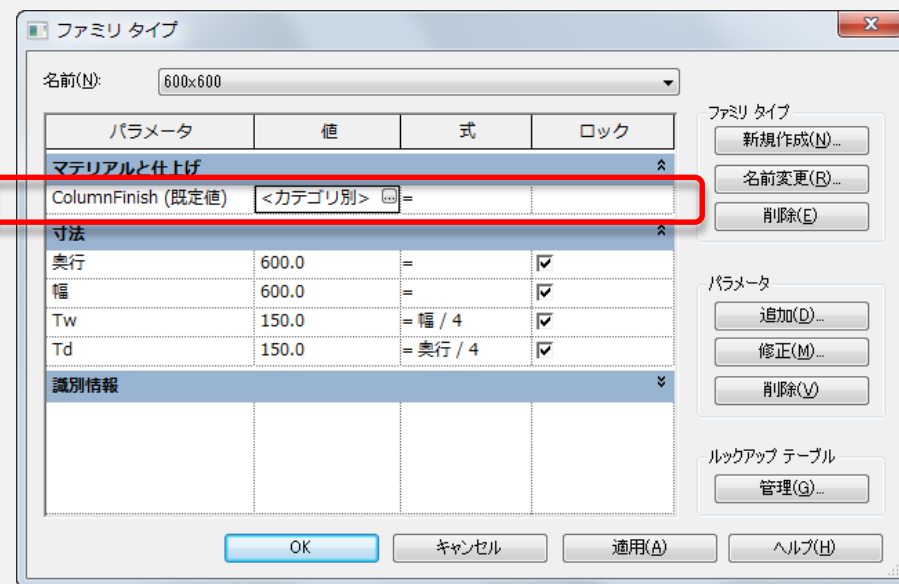
Sub AddParameter_Tw()
    '' add a parameter "Tw"
    Dim isInstance As Boolean = False
    Dim paramTw As FamilyParameter = _ m_familyMgr.AddParameter( _
        "Tw", BuiltInParameterGroup.PG_GEOMETRY, ParameterType.Length, isInstance)
    '' give initial values.
    Dim tw As Double = Utils.mmToFeet(150.0) '' in metric
    'Dim tw As Double = 0.5 '' in feet
    m_familyMgr.Set(paramTw, tw)
    '' add a formula (optional)
    m_familyMgr.SetFormula( _
        paramTw, "幅 / 4.0")
End Sub
```



3a.パラメータの追加

例: 柱の仕上げ

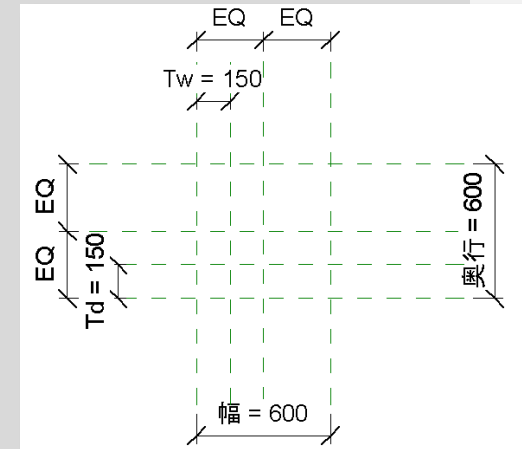
```
Sub AddParameter_Material()  
  
    ' add a parameter "Tw"  
    Dim param As FamilyParameter = m_familyMgr.AddParameter("Column Finish", _  
        BuiltInParameterGroup.PG_MATERIALS, ParameterType.Material, True)  
  
    ' we will come back to setting to a solid later  
End Sub
```



3b.寸法の追加

例: Tw

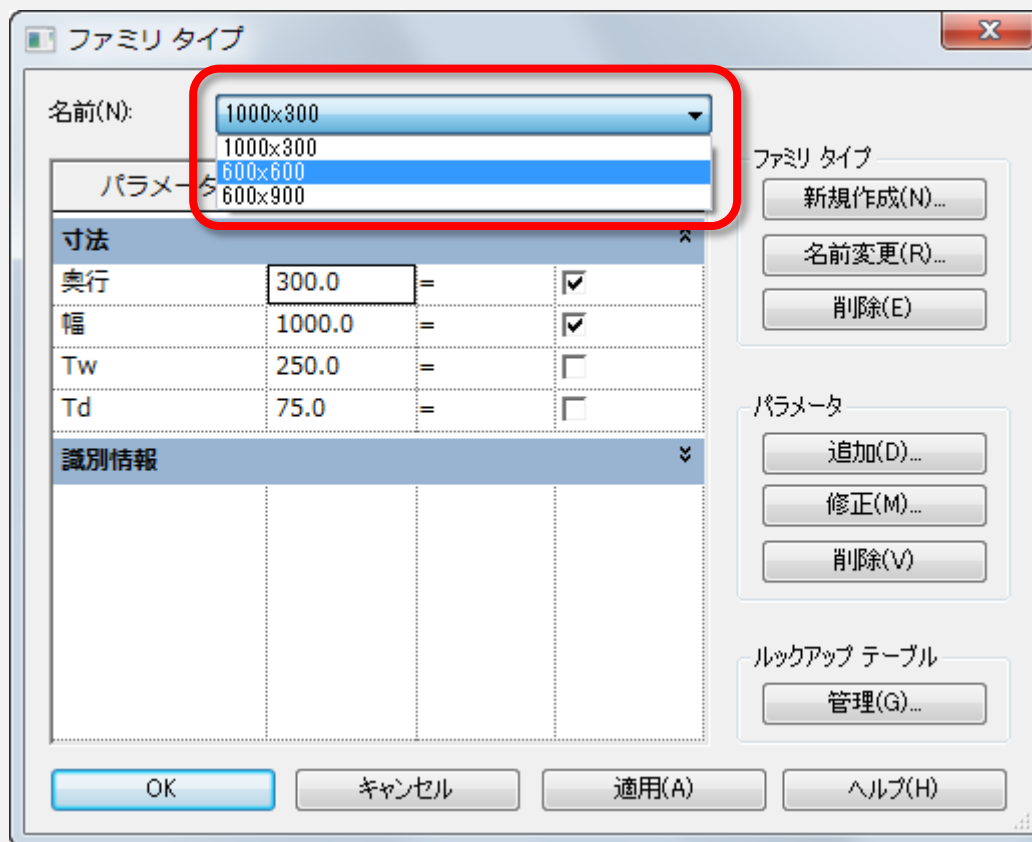
```
Sub AddDimention_Tw()  
    ' find the plan view that we want to place a dimension  
    Dim pViewPlan As View = _  
        Utils.FindElement(m_rvtDoc, GetType(ViewPlan), "下参照レベル")  
    ' find two reference planes which we want to add a dimension between  
    Dim ref1 As ReferencePlane = _  
        Utils.FindElement(m_rvtDoc, GetType(ReferencePlane), "左")  
    Dim ref2 As ReferencePlane = _  
        Utils.FindElement(m_rvtDoc, GetType(ReferencePlane), "OffsetV")  
    ' make an array of references  
    Dim pRefArray As New ReferenceArray  
    pRefArray.Append(ref1.GetReference())  
    pRefArray.Append(ref2.GetReference())  
    ' define a dimension line  
    Dim p0 As XYZ = ref1.FreeEnd  
    Dim p1 As XYZ = ref2.FreeEnd  
    Dim pLine As Line = Line.CreateBound(p0, p1)  
    ' create a dimension  
    Dim pDimTw As Dimension = _  
        m_rvtDoc.FamilyCreate.NewDimension(pViewPlan, pLine, pRefArray)  
    ' add label to the dimension  
    pDimTw.Label = m_familyMgr.Parameter("Tw")  
End Sub
```



4. 複数のホスト厚さタイプを追加

- テスト用の目的
- 実習の例ではホストなし

5.2つ以上のタイプを追加

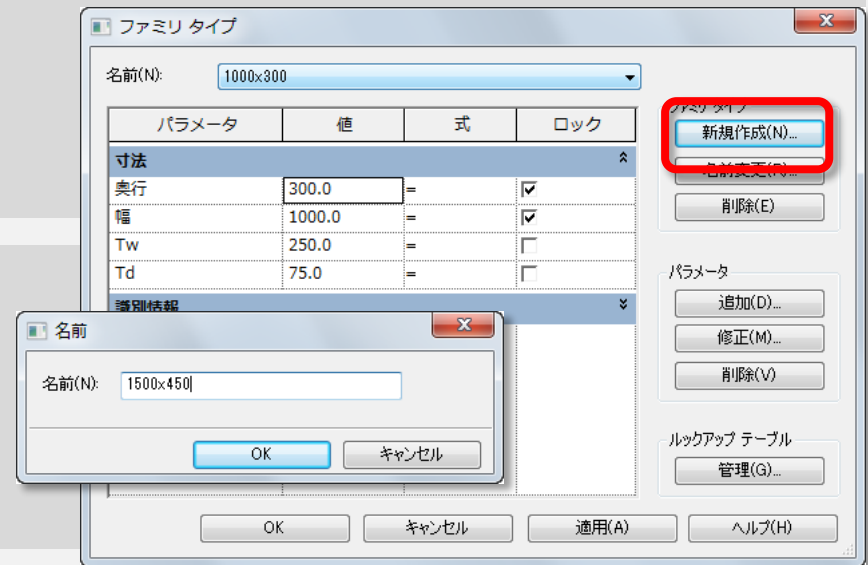


5. 2つ以上のタイプを追加

例: 幅 x 深さ

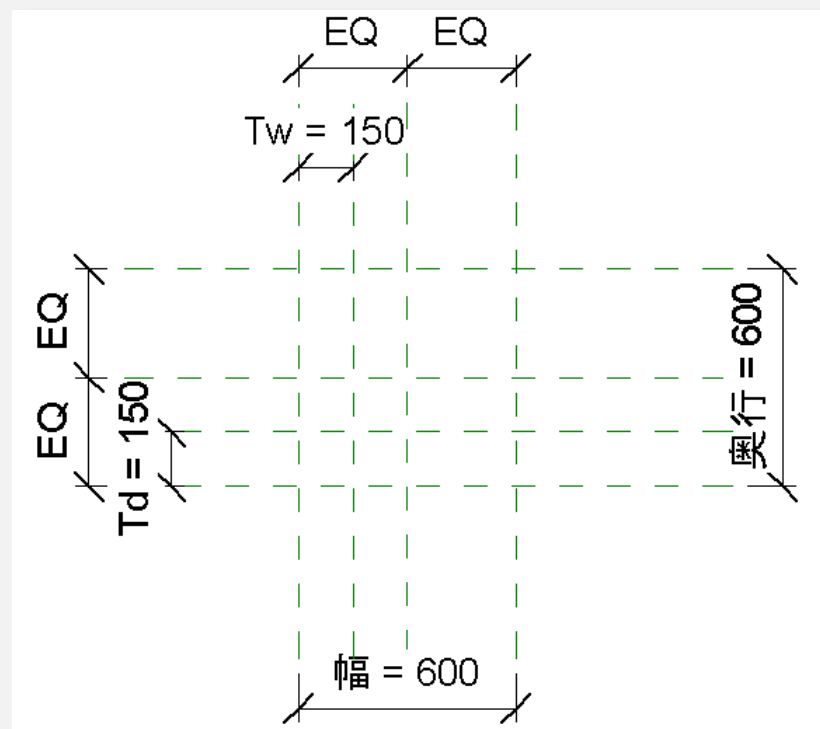
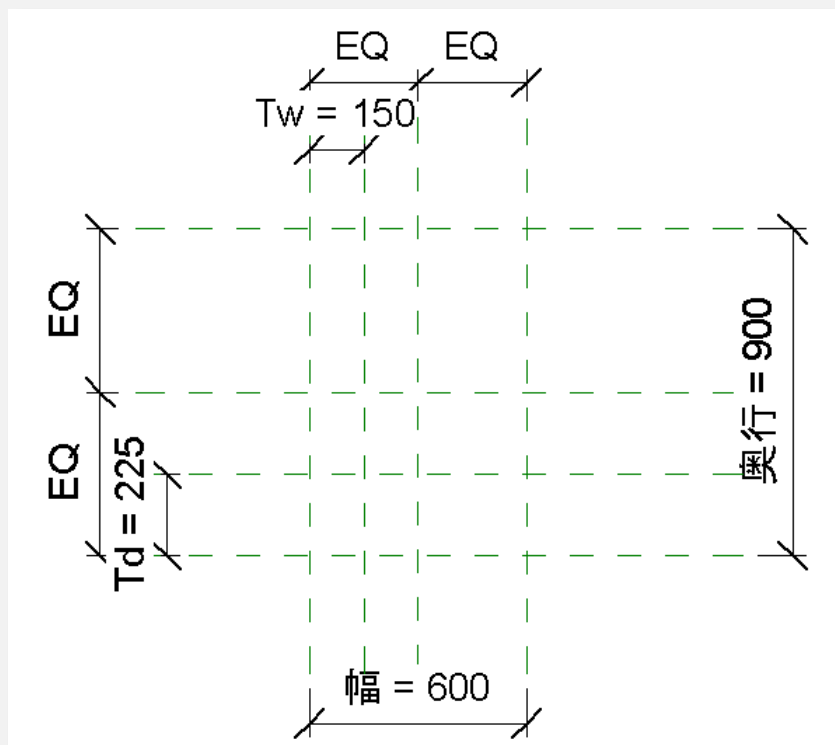
```
Sub AddType(ByVal name As String, ByVal w As Double, ByVal d As Double)
    ' add new types with the given name.
    Dim type1 As FamilyType = m_familyMgr.NewType(name)
    ' look for 'Width' and 'Depth' parameters and set them with the given values
    Dim paramW As FamilyParameter = m_familyMgr.Parameter("幅")
    Dim valW As Double = Utils.mmToFeet(w)
    If paramW IsNot Nothing Then
        m_familyMgr.Set(paramW, valW)
    End If
    Dim paramD As FamilyParameter = m_familyMgr.Parameter("奥行")
    Dim valD As Double = Utils.mmToFeet(d)
    If paramD IsNot Nothing Then
        m_familyMgr.Set(paramD, valD)
    End If
End Sub
```

```
Sub AddTypes()
    ' AddType(name,Width,Depth)
    AddType("600x900", 600.0, 900.0)
    AddType("1000x300", 1000.0, 300.0)
    AddType("600x600", 600.0, 600.0)
End Sub
```



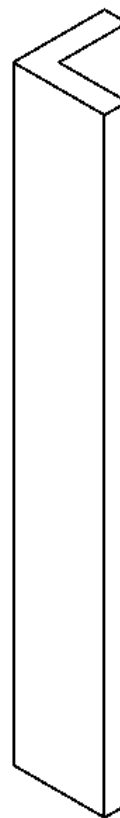
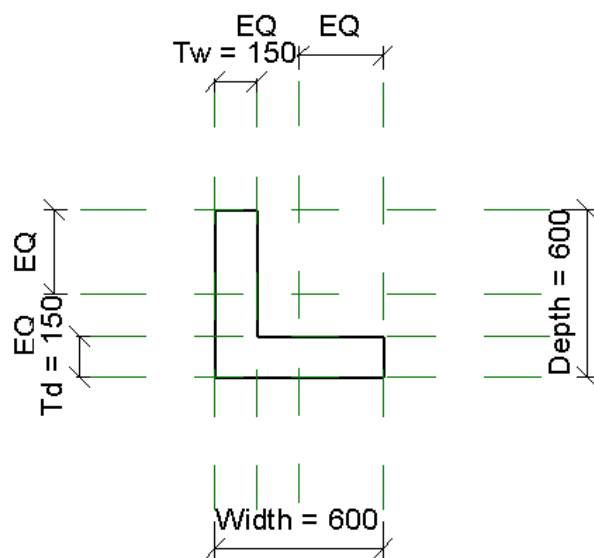
6. フレキシブル タイプとホスト(テスト用)

■ テスト手順



7. ジオメトリの単一レベルを追加

- ソリッドを追加
- 位置合わせを追加



7a. ジオメトリの単一レベルを追加

例: 押し出し

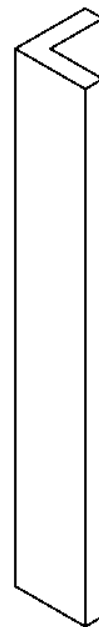
```
Function CreateSolid() As Extrusion
    '(1) define a simple L-shape profile
    Dim pProfile As CurveArrArray = CreateProfileLShape()

    '(2) create a sketch plane
    Dim pRefPlane As ReferencePlane = _
        Utils.FindElement(m_rvtDoc, GetType(ReferencePlane), "参照面")
    Dim pSketchPlane As SketchPlane = _
        m_rvtDoc.FamilyCreate.NewSketchPlane(pRefPlane.Plane)

    '(3) height of the extrusion. distance between Lower and Upper Ref Level.
    Dim dHeight As Double = Utils.mmToFeet(4000)

    '(4) create an extrusion here. at this point.
    Dim bIsSolid As Boolean = True ' as oppose to void.
    Dim pSolid As Extrusion = _
        m_rvtDoc.FamilyCreate.NewExtrusion(bIsSolid, pProfile, pSketchPlane,
        dHeight)
    Return pSolid

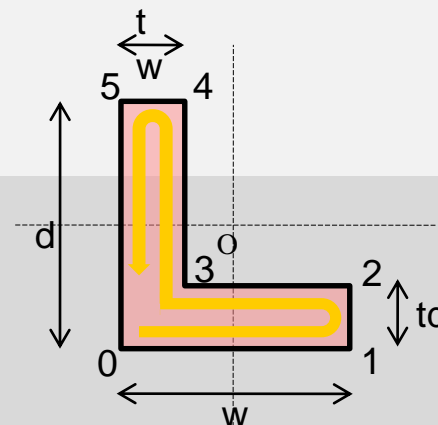
End Function
```



7a. ジオメトリの単一レベルを追加

例: L-形プロファイル

```
Function CreateProfileLShape() As CurveArrArray
    Dim w As Double = Utils.mmToFeet(600)
    Dim d As Double = Utils.mmToFeet(600)
    Dim tw As Double = Utils.mmToFeet(150)
    Dim td As Double = Utils.mmToFeet(150)
    '' define vertices (the last one is to make the loop simple)
    Const nVerts As Integer = 6 '' the number of vertices
    Dim pts() As XYZ = {New XYZ(-w / 2, -d / 2, 0), New XYZ(w / 2, -d / 2, 0), _
        New XYZ(w / 2, -d / 2 + td, 0), New XYZ(-w / 2 + tw, -d / 2 + td, 0), _
        New XYZ(-w / 2 + tw, d / 2, 0), New XYZ(-w / 2, d / 2, 0), _
        New XYZ(-w / 2, -d / 2, 0)}
    '' define a loop. define individual edges and put them in a curveArray
    Dim pLoop As CurveArray = m_rvtApp.Create.NewCurveArray
    Dim lines(nVerts - 1) As Line
    For i As Integer = 0 To nVerts - 1
        lines(i) = Line.CreateBound(pts(i), pts(i + 1))
        pLoop.Append(lines(i))
    Next
    '' then, put the loop in the curveArrArray as a profile
    Dim pProfile As CurveArrArray = m_rvtApp.Create.NewCurveArrArray
    pProfile.Append(pLoop)
    Return pProfile
End Function
```



7b. 位置合わせの追加

例: 参照面 “OffsetV” 上で面をロック

```
Sub AddAlignment_ReferencePlane(ByVal pSolid As Extrusion,
    ByVal normal As XYZ, ByVal nameRefPlane As String)

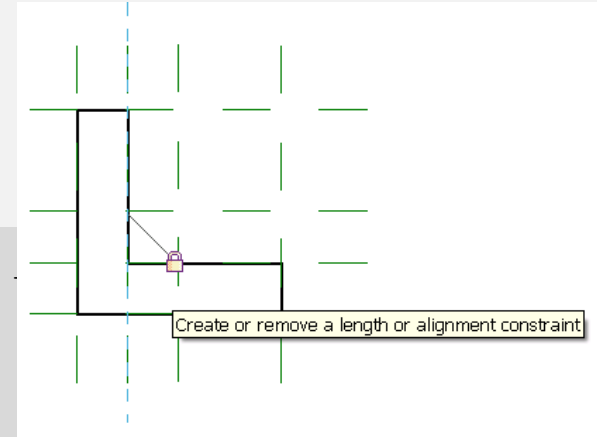
    ' get the plan view
    Dim pViewPlan As View = _
        Utils.FindElement(m_rvtDoc, GetType(ViewPlan), "下参照レベル")

    ' find reference planes
    Dim refPlane As ReferencePlane = _
        Utils.FindElement(m_rvtDoc, GetType(ReferencePlane), nameRefPlane)

    ' find the face of the solid
    Dim pFace As PlanarFace = Utils.FindFace(pSolid, normal, refPlane)

    ' create a locked alignment
    m_rvtDoc.FamilyCreate.NewAlignment( _
        pViewPlan, refPlane.GetReference(), pFace.Reference)

End Sub
```



7b.位置合わせの追加

例: レベルの位置合わせ

```
Sub AddAlignment_Level( _  
    ByVal pSolid As Extrusion, ByVal normal As XYZ, ByVal nameLevel As String)  
  
    ' which direction are we looking at?  
    Dim pView As View = Utils.FindElement(m_rvtDoc, GetType(View), "正面")  
  
    ' find the upper ref level. FindElement() is a helper function  
    Dim pLevel As Level = Utils.FindElement(m_rvtDoc, GetType(Level), nameLevel)  
  
    ' find the face of the box. FindFace() is a helper function.  
    Dim pFace As PlanarFace = Utils.FindFace(pSolid, normal)  
  
    ' create alignments  
    m_rvtDoc.FamilyCreate.NewAlignment(pView, pLevel.GetPlaneReference(),  
    pFace.Reference)  
  
End Sub
```

7b.位置合わせの追加

例: L-形ソリッドの位置合わせ

```
Sub AddAlignments(ByVal pSolid As Extrusion)

    AddAlignment_Level(pSolid, New XYZ(0.0, 0.0, 1.0), "上参照レベル")
    AddAlignment_Level(pSolid, New XYZ(0.0, 0.0, -1.0), "下参照レベル")
    AddAlignment_ReferencePlane(pSolid, New XYZ(1.0, 0.0, 0.0), "右")
    AddAlignment_ReferencePlane(pSolid, New XYZ(-1.0, 0.0, 0.0), "左")
    AddAlignment_ReferencePlane(pSolid, New XYZ(0.0, -1.0, 0.0), "正面")
    AddAlignment_ReferencePlane(pSolid, New XYZ(0.0, 1.0, 0.0), "背面")
    AddAlignment_ReferencePlane(pSolid, New XYZ(1.0, 0.0, 0.0), "OffsetV")
    AddAlignment_ReferencePlane(pSolid, New XYZ(0.0, 1.0, 0.0), "OffsetH")

End Sub
```

8. 満足が得られるまで 6 と 7 の繰り返し

- 6. フレキシブル タイプとホスト(テスト手順)
- 7. ジオメトリの追加

9. プロジェクト環境でのテスト

- テスト プロジェクトの作成

追加のクラスとメソッド

可視性

```
Sub SetVisibility(ByVal pSolid As Extrusion)
    ' set the visibility of the model not to shown in coarse.
    Dim pVis As FamilyElementVisibility = _
        New FamilyElementVisibility(FamilyElementVisibilityType.Model)
    pVis.IsShownInCoarse = False

    pSolid.SetVisibility(pVis)
End Sub
```

追加のクラスとメソッド

パラメータとの関連付け

```
Sub addMaterials(ByVal pSolid As Extrusion)
    ' get the material id that we are intersted in (e.g., "Glass")
    Dim pMat As Material = Utils.FindElement(m_rvtDoc, GetType(Material), "ガラス")
    Dim idMat As ElementId = pMat.Id
    ' add a parameter for material finish
    Dim paramFamilyMaterial As FamilyParameter = _
        m_familyMgr.Parameter("Column Finish")

    ' associate material parameter to the family parameter we just added
    Dim paramSolidMaterial As Parameter = pSolid.LookupParameter("マテリアル")
    m_familyMgr.AssociateElementParameterToFamilyParameter( _
        paramSolidMaterial, paramFamilyMaterial)

    ' let's add another type with Glass finish
    AddType("ガラス", 600.0, 600.0)
    m_familyMgr.Set(paramFamilyMaterial, idMat)
End Sub
```


ファミリ API SDK サンプル

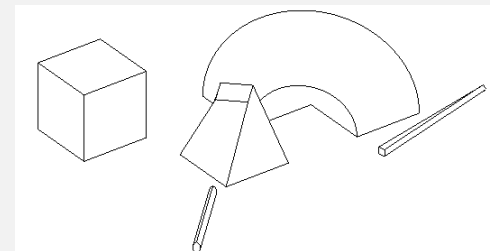
今後の参照用

ファミリ API SDK サンプル

学習リソース

<SDK パス>¥Samples¥FamilyCreation フォルダ

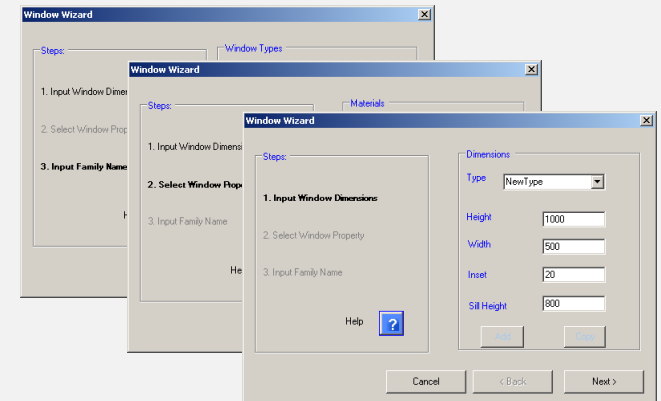
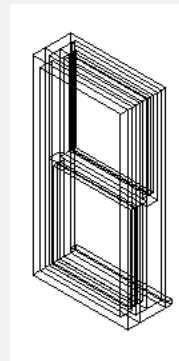
- AutoJoin
 - ファミリーモデリングとマスの使用に複数の一般フォームのジオメトリーを自動的に結合
- AutoParameter
 - ファミリドキュメントにファミリー、共有パラメーターを追加
- DWGFamilyCreation
 - DWG ファイルをファミリドキュメントに読み込み、インスタンスにタイプパラメーターを追加
- GenericModelCreation
 - 押し出し、ブレンド、回転、スイープを作成



ファミリ API SDK サンプル(続き)

学習リソース

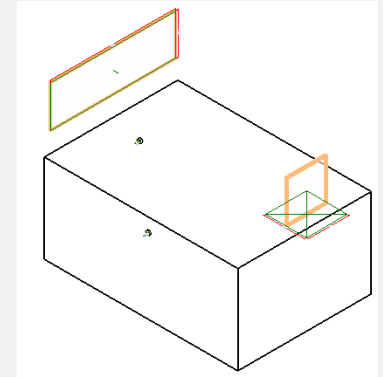
- TypeRegeneration
 - タイプ作成を試みて結果をレポート
- ValidateParameters
 - 全タイプでパラメーターの値をチェックして結果をレポート
- WindowWizard
 - ウィザードUIを実装してウィンドファミリーを作成



ファミリ API SDK サンプル(続き)

学習リソース

- CreateAirHandler – RME
 - パイプとダクト接続で空気処理機を作成
- CreateTruss – RST
 - トラスファミリドキュメントにモノトラスを作成



まとめ

- UI からの Revit ファミリ
 - ファミリとは？
 - どこから初めて、振る舞い、エディタ、何が可能なのか
 - ベスト プラクティス
- API を使ったファミリ作成
 - ベスト プラクティスに沿った習得
 - 例: L-形柱
 - 学習リソース

参考 Web ページ

- Revit API 開発者用ガイド(20xx を西暦年に置換してください)
 - <http://help.autodesk.com/view/RVT/20xx/JPN/?guid=GUID-F0A122E0-E556-4D0D-9D0F-7E72A9315A42>
- Revit 開発関連ブログ
 - Technology Perspective from Japan
 - http://adndevblog.typepad.com/technology_perspective/
 - The Building Coder, Jeremy Tammik's Revit API
 - <http://thebuildingcoder.typepad.com>
- ディスカッション グループ
 - <http://discussion.autodesk.com> > Revit Architecture > Revit API
- .NET 言語変換
 - Convert C# to VB.NET
 - <http://www.developerfusion.com/tools/convert/csharp-to-vb/>
 - Convert VB.NET to C#
 - <http://www.developerfusion.com/tools/convert/vb-to-csharp/>



Autodesk is a registered trademark of Autodesk, Inc., and/or its subsidiaries and/or affiliates in the USA and/or other countries. All other brand names, product names, or trademarks belong to their respective holders. Autodesk reserves the right to alter product and services offerings, and specifications and pricing at any time without notice, and is not responsible for typographical or graphical errors that may appear in this document.