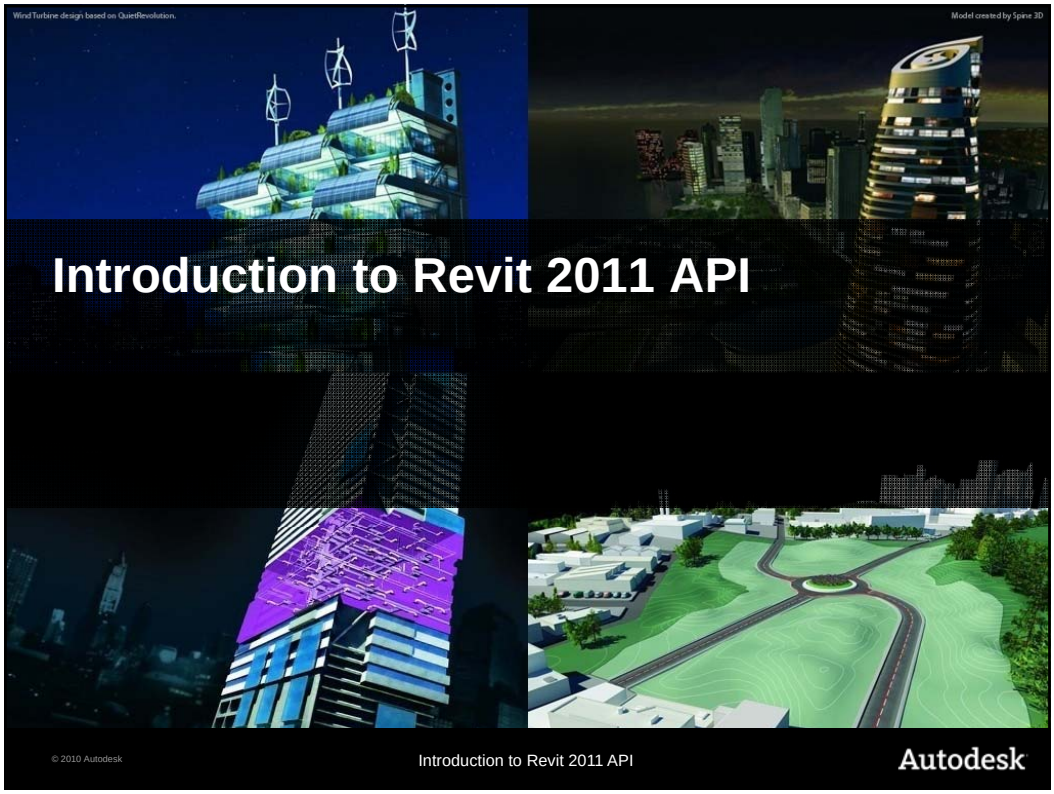


はじめる前に

LiveMeeting と音声会議 の使用方法

フルスクリーン　モード

フルスクリーンに拡張
ESC で解除

フィードバック

リアルタイム Q&A

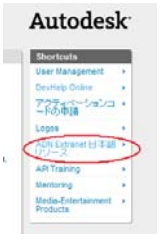
電話をミュートする　*6

© 2010 Autodesk
Introduction to Revit 2011 API
Autodesk

本日のWebcastは録画されます

近日中に録画とサンプルコードを以下の場所にポストします

http://adn.autodesk.com/ > 「ADN Extranet 日本語リソース 」



本日の内容

はじめに
- Revit 製品、Revit Software Development Kit

外部コマンドと外部アプリケーション
- 開発環境、外部コマンドとアプリケーション、アドインマニフェスト、開発支援ツール

Revit 要素
- 要素の識別
- 要素イテレート、フィルタ、LINQクエリ
- 要素の編集
- モデル作成

ユーザインターフェース
- リボン、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

はじめに

Revit 製品、 Revit Software Development Kit

Revit 製品

Revit 製品

- Revit Architecture
- Revit MEP (Mechanical, Electrical, Plumbing)
- Revit Structure

製品の配布方法

- DVD版 － ADNサイト
  - Software & Support > Revit > Downloads
  - RTMバージョン
- Web 更新バージョン －Autodesk ホームページ (www.autodesk.com)
  - Products > Autodesk Revit Architecture/MEP/Structure > Product Download>Data & Downloads （言語を選択） > Updates
  - Autodesk 製品サイトから更新バージョンがダウンロード可能

Revit Software Development Kit (SDK)

- SDKは製品に含まれています
  - インストーラの“インストールツールとユーティリティ”メニュー
  - Web バージョン
    <解凍フォルダ>¥support¥SDK¥RevitSDK.zip　（または Revit2011SDK.exe）
- Revit 2011 テンポラリーフォルダにSDKインストーラー
  - C:¥Autodesk¥RAC_2011_English_Win_32bit¥support¥SDK¥RevitSDK.exe
- Revit Developer Center に更新されたSDKをポスト
  - http://www.autodesk.com/developrevit

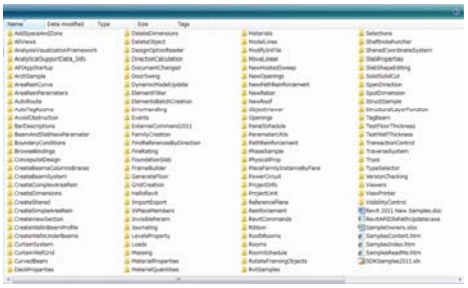
SDK の内容

- ドキュメント
  - Read Me First.doc （各コンテンツの説明）
  - Getting Started with the Revit API.doc （開発を始める方の為の基礎的な情報）
  - Revit Platform API Changes and Additions.doc （Revit 2011のＡＰＩの変更点と新機能について）
  - Revit 2011 API Namespace Remapping.xlsx （変更された名前空間の情報）
  - Revit 2011 API Developer Guide.pdf （Revit APIの解説書）
  - RevitAPI.chm （クラスリファレンス）
  - RevitAddInUtility.chm （アドインマニフェストファイルに アクセスを提供するクラスのリファレンス）
  - Autodesk Icon Guidelines.pdf （ユーザインターフェースのデザインのガイドライン）
  - Revit Structure （ 断面定義データとマテリアルプロパティ）
- ユーティリティツール
  - Add-In Manager （アドインロードツール）
  - RevitLookup （プロジェクトやファミリファイルの中身を見る）
- サンプル
  - Sample フォルダ （Revit APIで書かれたサンプルアドイン）
  - VSTA Samples フォルダ （VSTA のサンプル）

一度目を通す
常に使用
必要であれば

SDK のSamples フォルダ

- ドキュメント
  - SamplesReadMe.htm　（各サンプルプロジェクトの説明）
  - Revit 2011 New Samples.doc　（更新、または新規追加されたサンプル一覧）
- サンプル ユーティリティツール
  - RevitAPIDllsPathUpdater.exe　（Revit API DLLのパスの設定を変更）
- サンプルプロジェクトを統括するソリューションファイル
  - SDKSamples2011.sln
- 108種類のアドインの
  サンプル



Revit APIでRevit を拡張する方法

1. 外部コマンド
   - IExternalCommand インターフェースを実装
   - 「アドイン」リボンタブの「外部ツール」パネルのプルダウンに追加
   - アドインマニフェストファイルでRevitに登録

2. 外部アプリケーション
   - IExternalApplication インターフェースを実装
   - 「アドイン」リボンタブに専用のリボンパネルを追加
   - 外部コマンドを使用可能
   - アドインマニフェストファイルでRevitに登録

3. Visual Studio Tools for Application (VSTA) マクロ*) 機能拡張の予定なし
   - ２種類のマクロ: アプリケーションレベルとドキュメントレベル
   - 文法と機能は外部コマンドとほぼ同じ

Revit 2011 API の開発環境

- .NET APIを公開
  - 開発ツールはMicrosoft Visual Studio 2008
  - Microsoft .NET Framework 3.5 をサポート
  - プログラミング言語はC#、VB.NET、またはマネージド C++
  - Visual Studioプロジェクトの種類はクラスライブラリ
  - 参照するアセンブリ
    - <Revit インストールファルダ>¥Program¥RevitAPI.dll
    - < Revit インストールファルダ>¥Program¥RevitAPIUI.dll

外部コマンド

Hello World 外部コマンドを作成してアドインマニフェストで登録

外部コマンド

外部コマンド “Hello World” を作成

1. Visual Studioプロジェクトタイプは クラスライブラリ
2. 参照 するRevit API のアセンブリ
   - RevitAPI.dll
   - RevitAPIUI.dll
3. 名前空間
   - Autodesk.Revit.DB
   - Autodesk.Revit.UI
   - Autodesk.Revit.ApplicationServices
   - Autodesk.Revit.Attributes
4. IExternalCommandインターフェースのExecute() メソッドを実装
5. アドインマニフェストファイルでコマンドを登録

外部コマンドの作成

VB.NET の例

```
<VB.NET>
''  Hello World #1 - A minimum Revit external command.
<Autodesk.Revit.Attributes.Transaction(Autodesk.Revit.Attributes.TransactionMode.Manual)> _
<Autodesk.Revit.Attributes.Regeneration(Autodesk.Revit.Attributes.RegenerationOption.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("ご挨拶", "こんにちは！")

        Return Result.Succeeded

    End Function

End Class
</VB.NET>
```

外部コマンドの作成
C# の例

```
<C#>
// Hello World #1 - A minimum Revit external command.
[Autodesk.Revit.Attributes.Transaction(Autodesk.Revit.Attributes.TransactionMode.Manual)]
[Autodesk.Revit.Attributes.Regeneration(Autodesk.Revit.Attributes.RegenerationOption.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("ご挨拶", "こんにちは！");

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


外部コマンドの作成
IExternalCommand インターフェース

```
<VB.NET>
''  Hello World #1 - A minimum Revit external command.
<Autodesk.Revit.Attributes.Transaction(Autodesk.Revit.Attributes.TransactionMode.Manual)> _
<Autodesk.Revit.Attributes.Regeneration(Autodesk.Revit.Attributes.RegenerationOption.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("ご挨拶 ", " こんにちは ")

        Return Result.Succeeded

    End Function

End Class
</VB.NET>
```

1. IExternalCommand から派生

外部コマンドの作成
Execute() メソッド

```
<VB.NET>
''  Hello World #1 - A minimum Revit external command.
<Autodesk.Revit.Attributes.Transaction(Autodesk.Revit.Attributes.TransactionMode.Manual)> _
<Autodesk.Revit.Attributes.Regeneration(Autodesk.Revit.Attributes.RegenerationOption.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("ご挨拶 ", "こんにちは！")

        Return Result.Succeeded

    End Function

End Class
</VB.NET>
```

2. Execute() メソッドを実装

引数:
1st オブジェクトモデルにアクセス
2nd ユーザにコマンド異常終了のエラーメッセージ
3rd コマンド異常終了にハイライト表示する要素

戻り値:
- Succeeded
- Failed
- Cancelled

外部コマンドの作成
属性

```
<VB.NET>
''  Hello World #1 - A minimum Revit external command.
<Autodesk.Revit.Attributes.Transaction(Autodesk.Revit.Attributes.TransactionMode.Manual)> _
<Autodesk.Revit.Attributes.Regeneration(Autodesk.Revit.Attributes.RegenerationOption.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("ご挨拶 ", "こんにちは！")

        Return Result.Succeeded

    End Function

End Class
</VB.NET>
```

3. 属性を設定

トランザクションモード: トランザクションの振る舞いを制御
- Automatic
- Manual
- ReadOnly

描画モード:描画を制御
- Automatic
- Manual

外部コマンドの作成
属性

- 描画モード
  - IExternalCommand インターフェースや IExternalApplicationを実装するクラスに適用が必要
  - RegenerationOption.Automatic （自動）
    - モデルが変更される毎にRevitが再描画する
  - RegenerationOption.Manual （手動）
    - モデルが変更されてもRevitは再描画しない
    - Revit アドインは必要に応じてDocument.Regenerate() と Document.AutoJoinElements() でモデルの要素を更新する

外部コマンドの作成
属性

- トランザクションモード
  - IExternalCommandインターフェースを実装するクラスに適用が必要
  - TransactionMode.Automatic （自動）
    - Revitが外部コマンド実行前にトランザクションを作成し、終了後に戻り値によってコミット、またはロールバックする
  - TransactionMode.Manual （手動）
    - Revitは外部コマンドのトランザクションを作成しないが、トランザクショングループを作成し、エラー終了時はロールバックする
    - 外部コマンドより広域なトランザクションを作成し
    - Revit アドインはトランザクション、サブトランザクション、トランザクショングループを使用できる
  - TransactionMode.ReadOnly （読み込み専用）
    - トランザクションは作成されません。
    - 外部コマンドがトランザクションの作成やモデルの変更を試みると例外エラーが発生

外部コマンドの作成

“こんにちは！“の表示

```
<VB.NET>
''  Hello World #1 - A minimum Revit external command.
<Autodesk.Revit.Attributes.Transaction(Autodesk.Revit.Attributes.TransactionMode.Manual)> _
<Autodesk.Revit.Attributes.Regeneration(Autodesk.Revit.Attributes.RegenerationOption.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(“ご挨拶”, “こんにちは！”)

        Return Result.Succeeded

    End Function

End Class
</VB.NET>
```

4. ダイアログでメッセージを表示

タスクダイアログ:
Revit と同じメッセージボックスで
“こんにちは！”を表示

アドイン マニフェスト

外部コマンドの登録方法

- Revit 2011 のアドイン登録メカニズム
  - (Revit.iniを使用した登録は可能ですが、将来的にこの方法は削除の予定)
- Revit 起動時に自動的に読み込まれる
- ファイル拡張子は “.addin”
- アドインマニフェストファイルの配置場所

Windows XP

C:¥Documents and Settings¥All Users¥Application Data¥Autodesk¥Revit¥Addins¥2011¥

C:¥Documents and Settings¥<user>¥Application Data¥Autodesk¥Revit¥Addins¥2011¥

Vista / Windows 7

C:¥ProgramData¥Autodesk¥Revit¥Addins¥2011¥

C:¥Users¥<user>¥AppData¥Roaming¥Autodesk¥Revit¥Addins¥2011¥

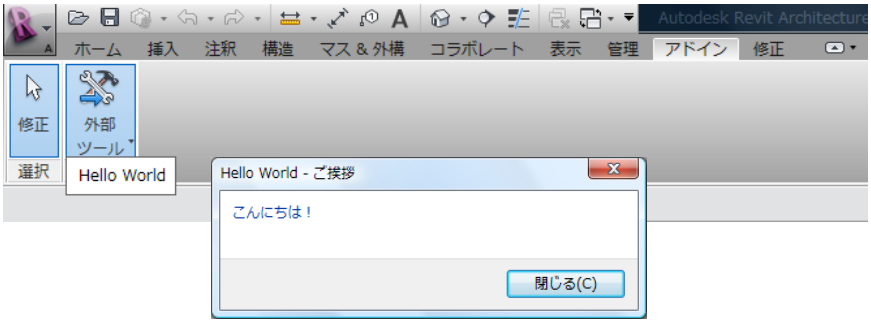
アドイン マニフェストファイル
ファイル形式

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<RevitAddIns>
  <AddIn Type="Command">
    <Text>Hello World</Text>
    <FullClassName>RevitIntroVB.HelloWorld</FullClassName>
    <Assembly>C:\RevitAPI 2011\RevitIntro\bin\HelloWorld.dll</Assembly>
    <AddInId>0B997216-52F3-412a-8A97-58558DC62D1E</AddInId>
  </AddIn>
</RevitAddIns>
```

含まれる情報:
- アドインタイプ: コマンドかアプリケーション
- [アドイン]タブの[外部ツール]パネル下の文字列
- 名前空間を含んだ完全なクラス名
- DLL のフルパス
- コマンドのGUID

外部ツールパネル
外部コマンドを実行

- 外部コマンドを登録すると 「アドイン」タブの「外部ツール 」にコマンド名が追加されます
- プルダウン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

外部アプリケーション

VB.NET の例

```
<VB.NET>
''  Hello World App - minimum external application
<Regeneration(RegenerationOption.Manual)> _
Public Class HelloWorldApp
    Implements IExternalApplication
    ''  OnShutdown() - called when Revit ends.
    Public Function OnShutdown(ByVal application As UIControlledApplication) _
    As Result _
    Implements IExternalApplication.OnShutdown
        Return Result.Succeeded
    End Function
    ''  OnStartup() - called when Revit starts.
    Public Function OnStartup(ByVal application As UIControlledApplication) _
    As Result _
    Implements IExternalApplication.OnStartup
        TaskDialog.Show("ご挨拶", "外部アプリケーションからこんにちは!")
        Return Result.Succeeded
    End Function
End Class
</VB.NET>
```

IExternalApplication を実装

OnShutdown はRevit 終了時に呼ばれる

OnStartup は Revit 起動時に呼ばれる

外部アプリケーション
C# の例

<C#>

```
// Hello World #3 - minimum external application
[Regeneration(RegenerationOption.Manual)]
public class HelloWorldApp : IExternalApplication
{
    // OnShutdown() - called when Revit ends.
    public Result OnShutdown(UIControlledApplication application)
    {
        return Result.Succeeded;
    }

    // OnStartup() - called when Revit starts.
    public Result OnStartup(UIControlledApplication application)
    {
        TaskDialog.Show("ご挨拶", "外部アプリケーションからこんにちは！");
        return Result.Succeeded;
    }
}
```

</C#>

外部アプリケーション
OnStartup メソッドで UIコンポーネントと外部コマンドの関連付け

<VB.NET>

```
'  set the information about the command we will be assigning to the button
    Dim pushButtonDataHello As New PushButtonData("PushButtonHello", "Hello World", m_assembly,
"RevitIntroVB.HelloWorld")
    '  add a button to the panel
    Dim pushButtonHello As PushButton = panel.AddItem(pushButtonDataHello)
    '  add an icon
    '  make sure you references to WindowsBase and PresentationCore, and import
System.Windows.Media.Imaging namespace.
    pushButtonHello.LargeImage = New BitmapImage(New Uri(m_imageFolder + "ImgHelloWorld.png"))
    '  add a tooltip
    pushButtonHello.ToolTip = "押しボタン"
```

</VB.NET>

外部アプリケーション
アドインマニフェストファイル

“Command” の代わりに
Type = “Application”
“Text”の代わりに“Name”

```
 <AddIn Type="Application">
    <Name>Hello World App</Name>
    <FullClassName>RevitIntroVB.HelloWorldApp</FullClassName>
    <Assembly>D:\RevitAPI 2011\Training\Labs\Revit Intro
Labs\RevitIntro\RevitIntroVB\bin\Debug\RevitIntroVB.dll</Assembly>
    <AddInId>08E8EFB1-FCC1-4b99-AD14-93523EE229AA</AddInId>
  </AddIn>
```


コマンドデータ

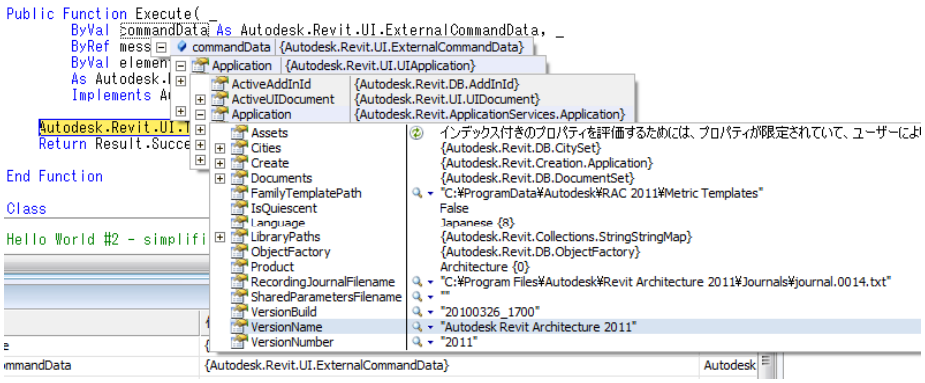
Revit オブジェクトモデルへの入口

コマンド データ

Revit オブジェクトモデルにアクセス

Executeメソッドの第 1 引数 ExternalCommandData

- Revit オブジェクトモデルにアクセスをするための入口



コマンド データ

Revit オブジェクトモデルにアクセス

- アプリケーション オブジェクト
  - アプリケーションオブジェクトは起動中のRevit
    - Autodesk.Revit.UI.UIApplication クラスはUIに関連した機能のアクセス
    - Autodesk.Revit.ApplicationServices.Application クラスはUI以外のアプリケーション全般のデータへのアクセス
- ドキュメント オブジェクト
  - ドキュメントオブジェクトはRevitで開かれたプロジェクトファイル
    - Autodesk.Revit.UI.UIDocument クラスはドキュメントのUI関連機能のアクセス
    - Autodesk.Revit.DB.Document クラスはドキュメントのモデルを格納するデータベース部分のアクセス

コマンド データ
Revit オブジェクトモデルにアクセス

- Revit のバージョンとドキュメントのタイトル名を取得

```
''  access to the version of Revit and the title of the document currently in use
Dim versionName As String = _
    commandData.Application.Application.VersionName
Dim documentTitle As String = _
    commandData.Application.ActiveUIDocument.Document.Title
```

- 壁の全てのタイプを取得

```
''  print out wall types available in the current rvt project
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にアクセス

UI 部分とDB部分

```
<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>
```

要素

Revit 要素の構成

要素

要素とは

- ドア、壁、寸法、ドアのタイプ、ビュー、マテリアル定義など建築物や図面のコンポーネント

要素

要素の分類　（その１）

- モデル要素
  - 例：壁、床、屋根、ドア、窓
- ビュー要素
  - 例：平面図、３Ｄ　ビュー、立面図、集計表
- グループ要素
  - 例：配列、グループ
- 注釈要素とデータ要素
  - 例：寸法、詳細線分、タグ、文字、記号
- スケッチ要素
  - 例：スケッチ平面、スケッチ、コンセプトマス
- 情報要素
  - プロジェクトデータ要素

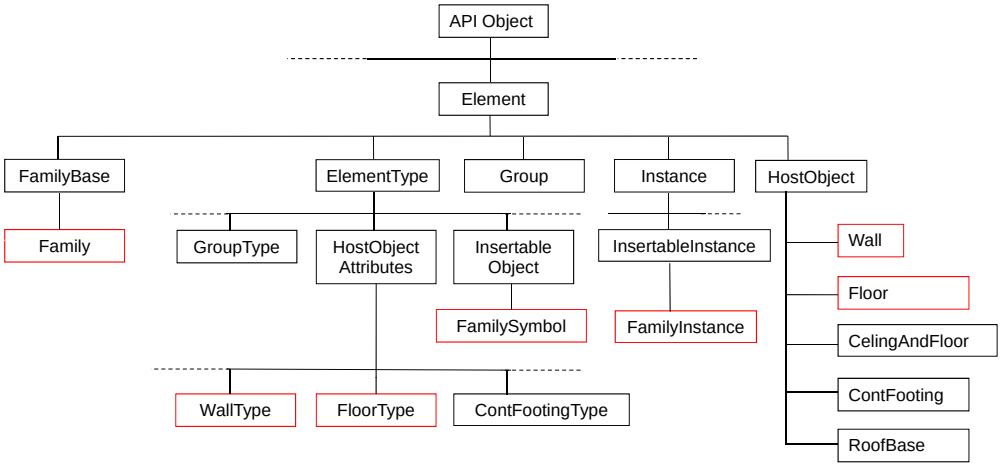
要素

要素の分類　（その２）

- カテゴリ
  - 例：“壁”、“床”、“屋根”、“ドア”、“窓”
- ファミリ
  - 例：“標準壁”、“片開”、“両開き窓”
- 要素タイプ
  - 例：“標準-150mm”、“w900h2000”、“w0800h1200”
- 要素インスタンス
  - 例：プロジェクトに配置された壁、床、屋根、ドア、窓

要素

モデル要素のクラスの派生図



要素

カテゴリ

	要素インスタンス		要素タイプ	
要素	**Element クラスから派生**	**カテゴリ**	**ElementType クラスから派生**	**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	長方形直線の壁開口部	< null >	---
床	HostObject/Floor	床	HostObjAttributes/FloorType	床
天井	< Element >	天井	HostObjAttributes	天井
屋根	HostObject/RoofBase/FootPrintRoof,ExtrusionRoof	屋根	HostObjAttributes/RoofType	屋根
柱	Instance/InsertableInstance/FamilyInstance	柱	InsertableObject /FamilySymbol	柱
コンポーネント(机)	Instance/InsertableInstance/FamilyInstance	家具	InsertableObject /FamilySymbol	家具
コンポーネント(木)	Instance/InsertableInstance/FamilyInstance	植物	InsertableObject /FamilySymbol	植物
階段	< Element >	階段	< ElementType >	階段
手すり	< Element >	手すり	< ElementType >	手すり
部屋	Room	部屋	< null >	---
部屋タグ	RoomTag	部屋タグ	< ElementType>	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	< ElementType >	< null >
文字	TextElement/TextNote	なし	LineAndTextAttrSymbol/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

要素

要素の識別方法

- システムファミリの要素インスタンスと要素タイプには定義クラスによって要素を識別することができます
- コンポーネントファミリの要素インスタンスはFamilyInstanceクラス、要素タイプはFamilySymbol クラスによって定義されます。この場合はカテゴリによって要素を識別することができます。

	システムファミリ	コンポーネントファミリ
要素タイプ	WallType クラス FloorType クラス	FamilySymbol クラス & カテゴリ
要素インスタンス	Wall クラス Floor クラス	FamilyInstance クラス & カテゴリ

要素

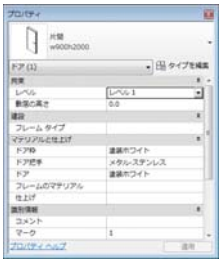
要素インスタンスを識別

```
<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 = "壁"
        ElseIf TypeOf elem Is Floor Then
            s = "床"
        ElseIf TypeOf elem Is RoofBase Then
            s = "屋根"
        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 = "ドア"
            ElseIf elem.Category.Id.IntegerValue = _
                BuiltInCategory.OST_Windows Then
                s = "窓"
            ElseIf elem.Category.Id.IntegerValue = _
                BuiltInCategory.OST_Furniture Then
                s = "家具"
            Else
                s = "コンポーネント　ファミリインスタンス"  '' e.g. Plant
            End If
      ...
</VB.NET>
```

要素

Parameters プロパティ

- Parameters プロパティはUIの要素プロパティに相当
- APIでは以下の方法でアクセス
  - Parameters プロパティ– 要素の全てのパラメータ
  - Parameter プロパティ– 指定したパラメータ



要素

Parameters プロパティ

```
<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 + vbCr + vbCr
        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 + vbCr
        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 = ""
        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>
```

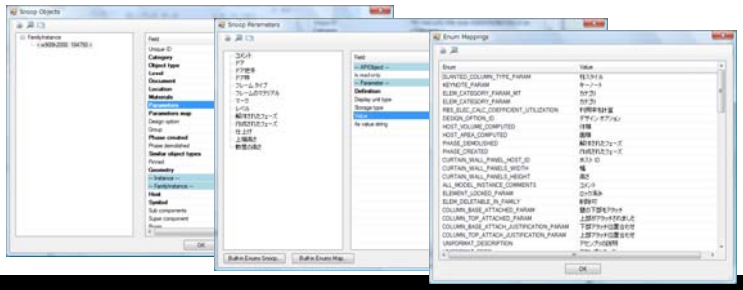

要素

Parameter プロパティ

指定のパラメータにアクセスする方法

- Parameter(**BuiltInParameter**) – 組み込みパラメータ列挙型
- Parameter(String) – パラメータ名
- Parameter(Definition) – パラメータ定義
- Parameter(GUID) – GUID

RevitLookup ツールを使って目的のパラメータがどのBuiltInParameter列挙型に相当するかを見つけることができます



要素

Parameter プロパティと 組み込みパラメータ ID

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

        Dim s As String = header + vbCr + vbCr
        ''  comments - most of instance has this parameter
        ''  (1) by name.  (Mark - most of instance has this parameter.)
        param = elem.Parameter("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>
```


要素

Location プロパティ

- 要素の位置 情報
- Location プロパティは2種類あります
  - LocationPoint – 座標点で示される位置 （例：家具）
  - LocationCurve – 曲線で示される位置 （例：壁）
- LocationPoint または LocationCurve クラスにキャストしてプロパティにアクセス

要素
位置 情報の取得

```
<VB.NET>
    ''  show the location information of the given element.
    Public Sub ShowLocation(ByVal elem As Element)
        Dim s As String = "位置情報: " + 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 + "終点(0)/始点 = " + PointToString(crv.EndPoint(0))
            s = s + "終点(1)/終点 = " + PointToString(crv.EndPoint(1))
            s = s + "長さ = " + crv.Length.ToString + vbCr
        End If
        …
    End Sub
</VB.NET>
```


要素
要素のジオメトリ

- ジオメトリ　オプション – 詳細レベルを指定(簡略 / 標準 / 詳細)
- ジオメトリの種類
  - ソリッド
  - ジオメトリインスタンス（窓やドアなど要素インスタンス）
  - 曲線
  - メッシュ
- ジオメトリはさらにソリッド、平面、エッジに細分化されている（ RevitLookupで確認できます）

- SDKサンプル
  - RevitCommands
  - ElementViewer
  - RoomViewer
  - AnalyticalViewer

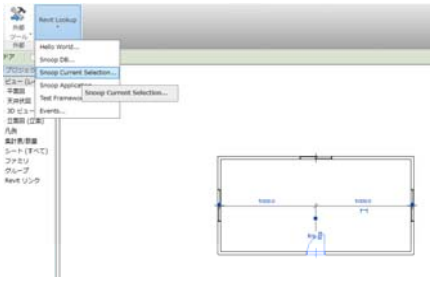
開発支援ツール

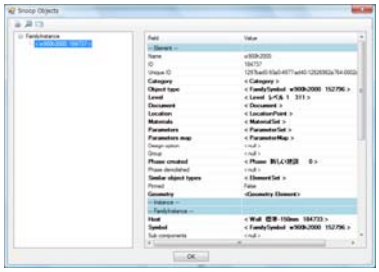
RevitLookup
Add-In Manager

開発支援ツール

RevitLookup ツール

- Revit データベースの中身を見るためのツール （Revit API プログラマには必需品）

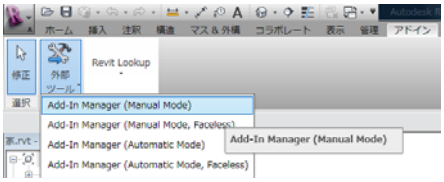


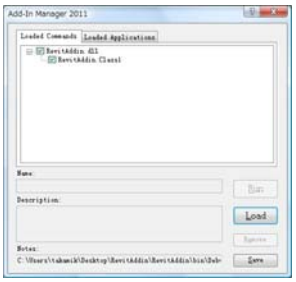


開発支援ツール

Add-In Manager

- アドインをRevit 実行中にロードやロード解除が可能
- 描画とトランザクションオプションが同じ場合に使用可能
  - （例：ManualとManualの場合は“Add-In Manager (Manual Mode)”コマンドを使用）
- “Faceless” コマンドは最後に実行したアドインを選択なしに実行





要素フィルタ

要素イテレート、要素フィルタ、LINQクエリ

要素フィルタ

要素の取得

- 要素は一次元の配列としてドキュメントに混在して格納されている
- イテレートやフィルタ（LINQクエリを併用）で目的の要素を取得する
- 一般的に頻繁に行われる処理
  1. 特定ファミリの全ての要素タイプを取得　（例：標準壁のタイプ）
  2. 定義クラスを指定して全ての要素インスタンスを取得 （例：全ての標準壁）
  3. タイプ名で要素タイプを取得 （例：“標準-150mm”のタイプ）
  4. 特定の要素インスタンスを取得（例：“レベル 1”）

要素フィルタ

システムファミリのタイプ

- 壁の全てのタイプを見つける

```
<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>
```

要素フィルタ

コンポーネントファミリのタイプ

- ドアの全てのタイプを見つける

```
<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>
```


要素フィルタ

システムファミリのインスタンス

- 全ての壁のインスタンスを見つける

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

要素フィルタ

コンポーネントファミリのインスタンス

- 全てのドアを見つける

```
<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>
```


要素フィルタ

システムファミリのタイプ

- 指定した壁のタイプを見つける

```
<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>
```

要素フィルタ

コンポーネントファミリのタイプ

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


要素フィルタ

要素タイプのインスタンス

- タイプを指定してその全ての要素インスタンスを見つける

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

要素フィルタ

クラスと要素名

- クラスと要素名で該当する要素を見つける

```
<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>
```


要素フィルタ

その他の要素フィルタクラス

その他フィルタクラス

- BoundingBoxContainsPointFilter
- ElementDesignOptionFilter
- ElementIsCurveDrivenFilter
- ElementIsElementTypeFilter
- ElementParameterFilter
- …

デベロッパガイドのセクション６（７７～８８ページ）やクラスリファレンスでご覧下さい

要素の編集

要素を編集する方法

要素の編集

要素を編集する方法

要素を編集する方法

- 要素のプロパティ、パラメータ、位置を変更
- Move やRotateなどドキュメントのメソッドで要素を変更

© 2010 Autodesk
Introduction to Revit 2011 API
Autodesk

要素の編集

要素 のタイプを変更

- 壁とドアのタイプを変更

```
<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-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), "片開", "w700h2000", _
         BuiltInCategory.OST_Doors)
    ''  assign a new family type.
    aDoor.Symbol = newDoorType
</VB.NET>
```


要素の編集

要素 のパラメータを変更

- 壁のパラメータ値を変更する

```
<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>
```

要素の編集

要素 の位置を変更

- 壁の位置を変更する

```
<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 = m_rvtApp.Create.NewLineBound(newPt1, newPt2)
    ''  change the curve.
    wallLocation.Curve = newWallLine
</VB.NET>
```


要素の編集

ドキュメント のメソッドで要素を変更

- 要素を移動と回転する

```
<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 = m_rvtApp.Create.NewLineBound(pt1, pt2)
        m_rvtDoc.Rotate(elem, axis, Math.PI / 12.0)
</VB.NET>
```

要素の編集

グラフィックの再描画

- RegenerationOption.Manual 属性を設定した場合、モデルのジオメトリ変更が伴う要素の編集で、変更されたジオメトリにアクセスする場合は、事前にDocument.Regenerateメソッドでグラフィックの再描画が必要

```
m_rvtDoc.Regenerate()
```

- RegenerationOption.Automatic 属性を設定した場合、Revitが必要の応じて自動的に再描画する

モデルの作成

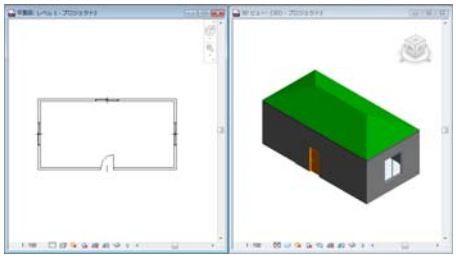
要素インスタンスの作成方法

モデルの作成

要素の作成

- ジオメトリ オブジェクトの新規作成:
  Application.Create.NewXxx()　例： NewLineBound()、XYZ
- モデル要素の新規作成:
  Document.Create.NewXxx()　例：NewWall(), NewFamilyInstance()

- 要素のタイプや作成条件に応じてオーバーロード関数を用意
  例：５種類の NewWallメソッド、９種類のNewFamilyInstanceメソッド



モデルの作成

壁の作成

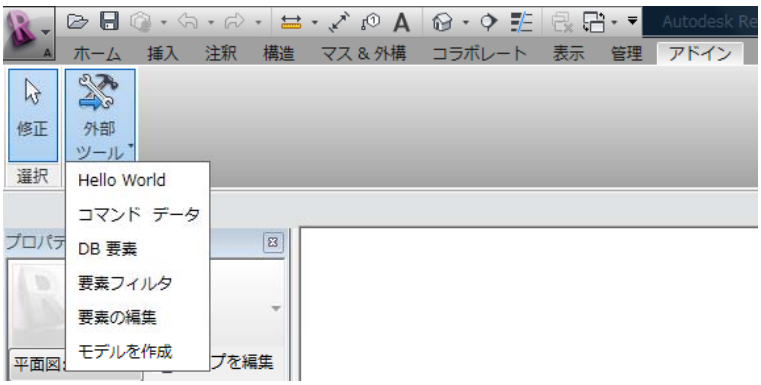
```
<VB.NET>
''  create walls
Sub CreateWalls()
    ''  get the levels we want to work on.
    Dim level1 As Level = ElementFiltering.FindElement(m_rvtDoc, GetType(Level), "レベル 1")
    Dim level2 As Level = ElementFiltering.FindElement(m_rvtDoc, GetType(Level), "レベル 2")

    ''  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 = m_rvtApp.Create.NewLineBound(pts(i), pts(i + 1))
        ''  create a wall using the one of overloaded methods.
        Dim aWall As Wall = m_rvtDoc.Create.NewWall(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>
```

モデルの作成

ドアの作成

```
<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), _
          "片開", "w900h2000", 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 API Intro サンプル



Revit UI

リボン、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

グラフィックユーザインタフェース のカスタマイズ

リボン API

概要

- GUI のカスタマイズはリボン API でのみ可能
  - 旧メニューとツールバーはリボンに移行が必要
- WPF の知識は必要なし
- GUI カスタマイズのガイドライン
  - Ribbon design guidelines.pdf
  - Autodesk Icon Guidelines.pdf
- 今後拡張の予定

リボン API

概要

- カスタマイズリボンパネルをアドインタブと解析タブに追加可能
- カスタムリボンタブは追加不可
- 外部ツールパネルに外部コマンド
- 外部アプリケーションはカスタムリボンパネル
  - ボタン
  - プルダウン リスト ボタン
  - スタックされたボタン
  - 分割ボタン
  - ラジオグループ
  - コンボボックス
  - 文字ボックス
  - 引き出しパネル

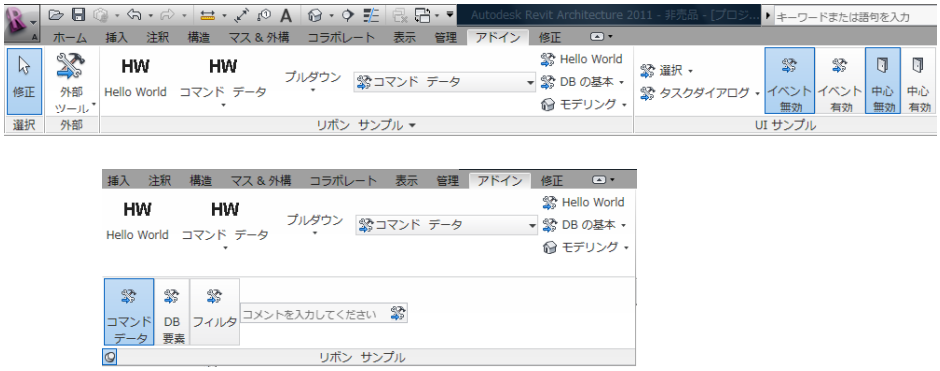
リボン API

クラス

- RibbonPanel
  - アドイン と解析リボンパネル
- RibbonItem
  - ボタン、コンボボックス、文字ボックス、ラジオボタンなど
- PushButton, PushButtonData
  - ボタン情報を管理
- PulldownButton, PulldownButtonData
  - ドロップダウン リスト ボタン情報を管理
- SplitButton, SplitButtonData
  - 分割ボタン情報を管理
- ComboBox, ComboBoxData
  - コンボボックス情報を管理

Revit API UI サンプル

リボン



ユーザ選択

座標点とオブジェクトを選択

ユーザ選択

概要

- 任意の座標点、要素（単一 / 複数)
  - 要素の点、エッジ、面
- メソッド
  - PickObject　（単一要素選択）
  - PickObjects　（複数要素選）
  - PickElementsByRectangle　（ボックス使った複数要素選択 ）
  - PickPoint　（アクティブな作業平面上の座標点の選択）

```
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.Elements.Size;
    string prompt = string.Format("{0} elements added to Selection.",
    newSelectionCount - selectionCount);
    TaskDialog.Show("Revit", prompt);
}
```

ユーザ選択

スナップの種類

- 選択時のスナップの種類を定義が可能

```
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可能
  - View.SketchPlane()

ユーザ選択

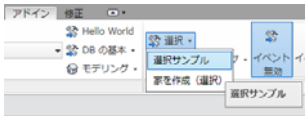
選択フィルタ

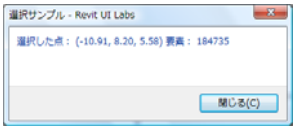
- 選択時に ISelection インタフェースでオブジェクトをフィルタ
  - AllowElement メソッド　（要素のフィルタ）
  - AllowReference メソッド　（ジオメトリオブジェクトのフィルタ）

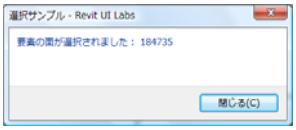
```
public void SelectPlanarFaces(Autodesk.Revit.DB.Document document)
{
        UIDocument uidoc = new UIDocument(document);
        ISelectionFilter selFilter = new PlanarFacesSelectionFilter();
        IList<Reference> faces = uidoc.Selection.PickObjects(ObjectType.Face, selFilter, "Select multiple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

Revit API UI サンプル

選択







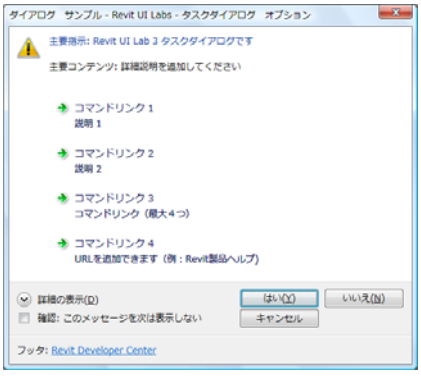
タスクダイアログ

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

- モーダルダイアログボックス
- Revit スタイルのメッセージボックス
- 以下の目的で使用
  - 情報を表示
  - ユーザに処理を選択させる

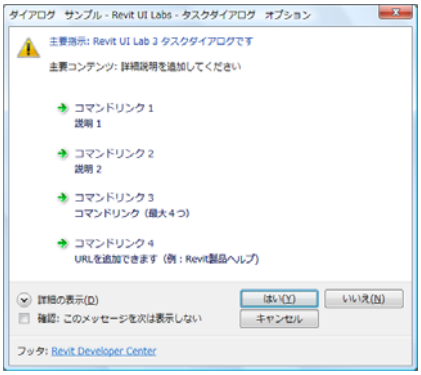


タスクダイアログ

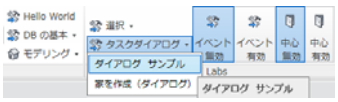
概要

- タスクダイアログを作成する２つの方法
  - TaskDialogインスタン作成、プロパティの設定、Showメソッドを呼ぶ
    - Autodesk.Revit.UI.**TaskDialog**
  - Show 静的メソッドを実行

- プロパティの設定
  - 指示
  - 詳細説明
  - アイコン
  - ボタン
  - コマンドリンク
  - 検証結果
  - その他

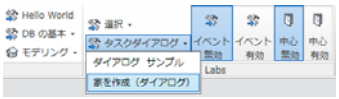


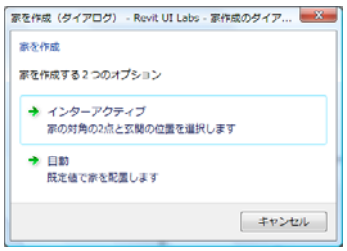
Revit API UI サンプル
タスクダイアログ




```
<CS>
    //' (0) create an instance of task dialog to set more options.
    TaskDialog myDialog = new TaskDialog("Revit UI Labs - タスクダイアログ  オプション");

    //' (1) set the main area. these appear at the upper portion of the dialog.
          //'
    myDialog.MainIcon = TaskDialogIcon.TaskDialogIconWarning;
    //' or TaskDialogIcon.TaskDialogIconNone.
    myDialog.MainInstruction = "主要指示: Revit UI Lab 3 タスクダイアログです";
    myDialog.MainContent = "主要コンテンツ: 詳細説明を追加してください";

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


Revit API UI サンプル
ダイアログで家を作成






```
<CS>
    TaskDialog houseDialog = new TaskDialog("Revit UI Labs - 家作成のダイアログ");
    houseDialog.MainInstruction = "家を作成";
    houseDialog.MainContent = "家を作成する２つのオプション";
    houseDialog.AddCommandLink(TaskDialogCommandLinkId.CommandLink1, "インターアクティブ", "家の対角の2点と玄関の位置を選択します");
    houseDialog.AddCommandLink(TaskDialogCommandLinkId.CommandLink2, "自動", "既定値で家を配置します");
    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 のイベント

イベント

概要

- 起きた事を伝えるための“通知”
- .NET イベント標準に準拠
  - ‘Pre’ （事前）と ‘Post’ （事後）イベント
- 種類
  - アプリケーション 全体
  - ドキュメント 単位

イベント

概要

- データベース 関連とUI 関連イベントに分類
  - データベース 関連イベントは Application とDocument クラスで公開
  - UI 関連イベントは UIApplication クラスで公開
- イベント時の編集モードを確認
  - Document.IsModifiable プロパティ
  - Document.IsReadOnly プロパティ
- 'pre'（事前）イベントで通知された処理をキャンセル可能
  - RevitEventArgs.Cancellable プロパティ
  - RevitAPIPreEventArgs.Cancel プロパティ

イベント

イベントハンドラ関数、イベント登録と登録解除

- イベントハンドラ関数

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

}
```

- イベント登録

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

- イベント登録解除

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

イベント
2011の変更

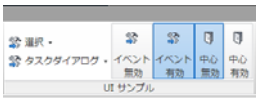
- 名前空間の変更
  - Autodesk.Revit.DB.Events
  - Autodesk.Revit.UI.Events
- 新しいイベント
  - Applicatoin.DocumentChanged
    - 追加、変更、削除された要素ID
  - UIApplication.Idling

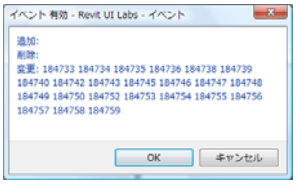
イベント
2011の変更

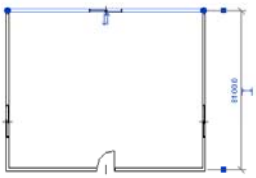
- イベントでトランザクションは自動的に作成されません

Revit API UI サンプル

イベント







```
//' register the document changed event
application.ControlledApplication.DocumentChanged += UILabs_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 = "追加: ";
foreach (ElementId id in idsAdded)
{
   msg += id.IntegerValue.ToString() + " ";
 }
```


ダイナミックモデルアップデート

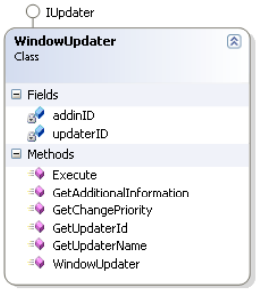
概要

- モデル変更時に更にモデルを編集可能
- 要素の追加、変更、削除を監視

ダイナミックモデルアップデート

アップデータオブジェクト

- アップデータ オブジェクト
  - 要素を監視して変更する
  - IUpdater インターフェースを実装
- **IUpdater** インタフェースのメソッド
  - 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

- アップデータの登録
  - OnStartUp はアプリケーション全体
  - ExternalCommand はコマンド単位

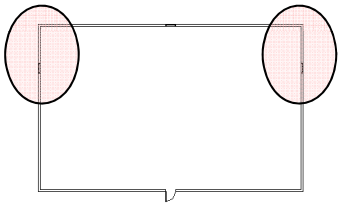
```
WindowUpdater updater = new WindowUpdater(application.ActiveAddInId );
// Register the updater in the singleton UpdateRegistry class
UpdaterRegistry.RegisterUpdater( updater );
```

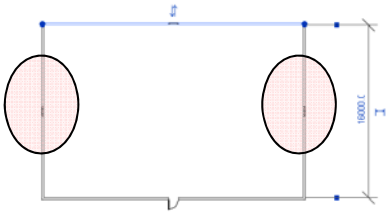
- トリガの追加
  - モデル変更の範囲 – 要素IDや要素フィルタで要素を指定
  - モデル変更の種類 – 追加、削除、変更（ジオメトリ、パラメータ、プロパティ）

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

Revit API UI サンプル
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 menifest.
//' 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());
```





その他 Revit リソース

- 製品オンラインヘルプ
- ディスカッショングループ（英語）
  - http://discussion.autodesk.com > Revit Architecture > Revit API
- The Building Coder - Jeremy Tammik Revit API ブログ (英語)
  - http://thebuildingcoder.typepad.com
- Autodesk Developer Network
  - http://www.autodesk.co.jp （パートナー > ADN > OTW > ADN)
  - 質問窓口（日本語と英語）
  - テクニカルノート
  - Webcast 録画

ご質問は？（Q&A）

ご参加ありがとうございました


Autodesk®