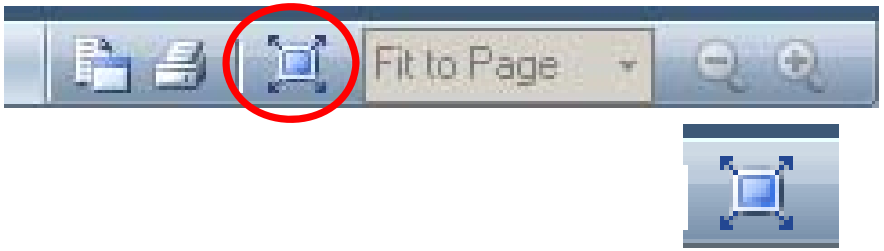
Revit 2011 API 介绍

叶雄进 Joe Ye

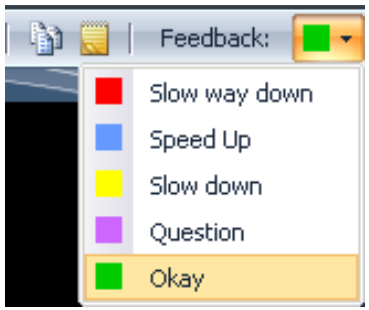
Developer Technical Services

如何使用LiveMeeting and conference 电话

全屏模式

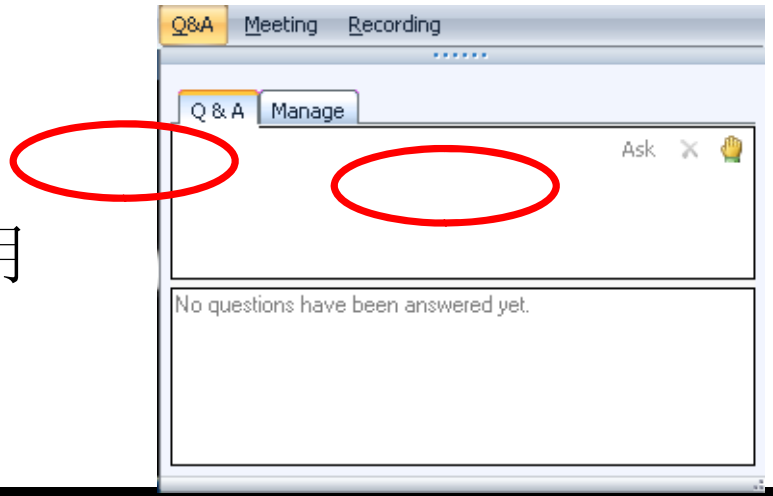


速度反馈



在线提问

课程结束后，可以用电话提问



关于培训者

叶雄进 Joe Ye
Joe.Ye@autodesk.com

5年工程软件开发
3年软件开发咨询

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

本次培训目的

本次培训目的

- 2011版API新功能的用法

不包含Revit2011基本详细的开发知识

- Revit 2011基础详细的开发技术课程从Developer Center下载
  - http://download.autodesk.com/media/adn/Revit_2011_API_DevTV_Chinese.zip
  - 培训中包含大量的演示，容易理解和学习。
    - Revit 二次开发入门
    - Revit 二次开发的完整过程
    - 遍历Revit内部对象
    - 访问和编辑对象的参数
    - 调试工具RevitLookup

今日培训实况可下载...

今日培训过程将录制，课程结束后几天内发布到网上:

ADN 网站

http://adn.autodesk.com
Revit > Knowledgebase > Whitepapers and Training Videos

官网开发者中心（Developer Center）

API Training & Consulting > Schedule
http://www.autodesk.com/developer

调查

1. 工作描述？
2. 对Revit Architecture、Structure、MEP产品的掌握程度？
3. 对Revit二次开发掌握熟悉程度?
4. 这次培训对你有何意义？

内容提纲

- Revit二次开发起步
  - 产品和SDK介绍
  - Revit 二次开发环境
  - Revit 二次开发的完整过程
- Revit数据库对象
  - 数据库对象特征
  - 过滤数据库中对象
  - 访问对象的参数
  - 修改，删除，创建Revit对象
- Revit 2011图形用户界面交互
  - Ribbon 界面编程
  - 选择集应用
  - 任务对话框
  - 事件和模型动态更新

二次开发起步

产品， SDK 和 加载dlls

Revit 二次开发所需条件

- 三个条件：
  - Revit 产品
  - Revit SDK
  - 开发工具

Revit 产品线

- Revit三个系列产品
  - Revit Architecture，用于建筑设计
  - Revit Structure ，用来建立结构模型
  - Revit MEP(**M**echanical, **E**lectrical, **P**lumbing), 用于建筑设备设计

- 如何获得软件安装包
  - DVD 版本在ADN网站可以获得
    - Software & Support > Revit > Downloads
    - 现可下载中文版Revit 2011 产品
  - 在Autodesk网站可以下载
    - Products > Revit Architecture, MEP, Structure > Product Download
    - 下载最新的软件
    - 具有独立的Server Pack，在已经安装的软件上更新。

Revit SDK

获取Revit SDK途径

- 产品安装包/安装光盘中有SDK
  - ..\support\SDK\RevitSDK.zip, 在安装光盘中搜索“SDK”可找到
- ADN网站
  - Software & Support > Revit > Downloads
  - 链接 http://adn.autodesk.com/adn/servlet/index?siteID=4814862&id=5017413&linkID=4901650
- Autodesk网站最新下载
  - http://www.autodesk.com/developer -->Products & Technologies--> Revit
  - http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=2484975

SDK中资源介绍

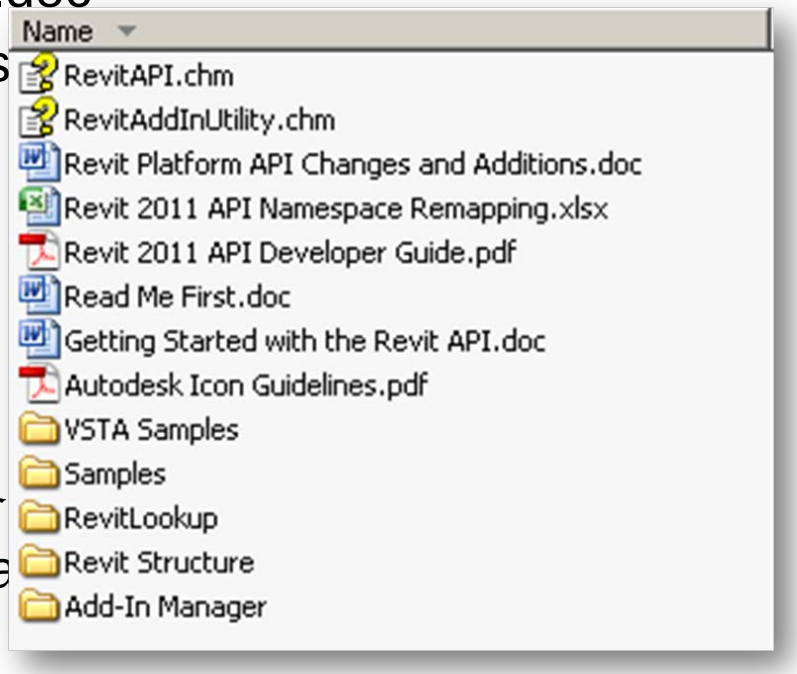
说明性文档

- Read Me First.doc
- Getting Started with the Revit API.doc
- Revit Platform API Changes and Additions.doc
- Revit 2011 API Namespace Remapping.xls

开发参考文档

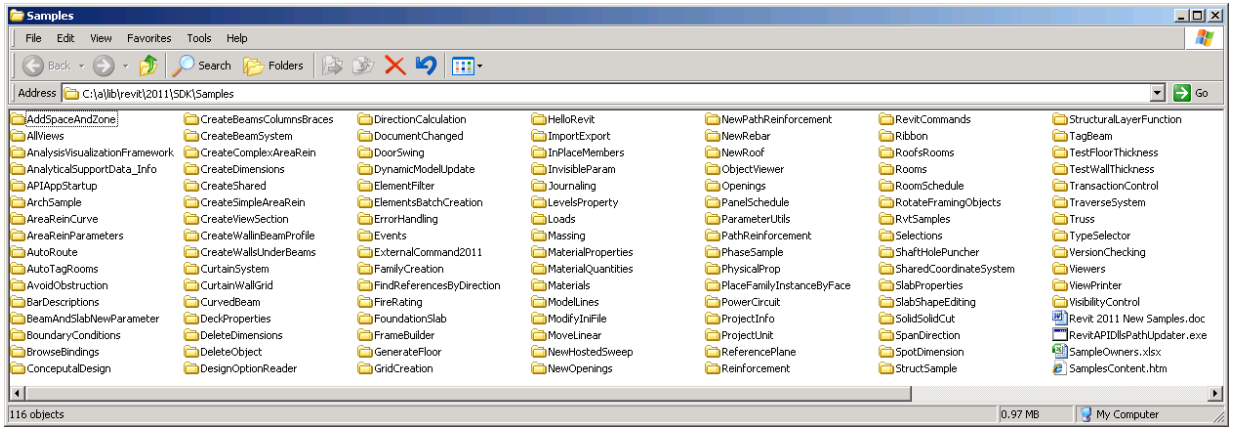
- RevitAPI.chm
- Revit 2011 API Developer Guide.pdf

- 需要时阅读
  - RevitAddInUtility.chm – 制作安装程序时
  - Autodesk Icon Guidelines.pdf – 制作用户界
  - Revit Structure – section definitions and ma
- 重要工具
  - Add-In Manager
  - RevitLookup
- VSTA Samples



SDK中源代码工程

- VSTA Samples
- 例程代码
  - 都在Samples 文件夹中
  - 包含110多个例子
  - 基本涵盖了Revit的API用法
- 工具和文档
  - RevitAPIDllsPathUpdater.exe
  - SamplesReadMe.htm
  - SDKSamples2011.sln



Revit API 开发工具和语言

- 两种开发工具
  - Visual Studio
    - 条件
      - Microsoft Visual Studio 2008
      - Microsoft .NET Framework 3.5
      - 语言：C# 、VB.NET、Managed C++
      - 引用接口组件装配 RevitAPI.DLL, RevitAPIUI.dll
  - VSTA （Visual Studio Tools for Application)
    - 使用Revit自带的开发环境
    - 用C#，VB.Net 语法
    - 生成宏

Revit API 接口装配Dll

- Revit API 接口装配dll在每个产品安装目录中
  - RevitAPI.dll
  - RevitAPIUI.dll
- 从Revit 2011开始，Dll 分成两个文件， DB 和UI 各一个文件. 之前只有RevitAPI.dll 一个文件
- Revit Architecture, Structure and MEP 包含相同接口装配dll
  - 只要版本相同，三个产品的接口文件一样
  - 如果对Revit Structure开发，可以引用MEP中的接口装配dll
  - 但是有些接口只适用于特定的产品。比如使用了荷载类的二次开发程序，只能在Revit Structure正确运行。

开发起步

最简单命令的第一步：外部命令和插件加载文件

用 **Visual Studio** 开发**Revit**

- 两种方式来扩展Revit的功能
  - **方式一：外部命令 （External Command）**
    - 功能：添加一个Revit命令
    - 描述: 由用户点击按钮来启动命令
    - 使用最频繁
  - **方式二：外部应用（External Application）**
    - 功能：可以添加菜单和工具条,或其它初始化命令
    - 描述：在启动和关闭Revit.exe时自动执行
    - 一般会用到，用量不多

外部命令

- 功能： 运行命令后，弹出消息框，显示Hello World

创建外部命令步骤

1. 新建一个 类库/Class Library 工程
2. 引用Revit接口装配Dll ： RevitAPI.DLL 和 RevitAPIUI.DLL
   - 将Copy Local 属性设置为False
3. 命名空间引用，如：
   - using Autodesk.Revit;
   - using Autodesk.Revit.DB;
   - using Autodesk.Revit.UI;
   - …
4. 为命令类加属性
   - [Transaction(TransactionMode.Automatic)]
   - [Regeneration(RegenerationOption.Automatic)]
5. 新建类从IExternalCommand派生
6. 重载Execute()方法
7. 在Execute中添加代码来实现命令功能

实现外部命令

最小的VB.Net外部命令

```
<VB.NET>
''  Hello World #1 - A minimum Revit external command.
<Autodesk.Revit.Attributes.Transaction(Autodesk.Revit.Attributes.TransactionMode.Automatic)> _
<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("My Dialog Title", "Hello World!")

        Return Result.Succeeded

    End Function

End Class
</VB.NET>
```

实现外部命令

最小的C#外部命令

```
<C#>
// Hello World #1 - A minimum Revit external command.
[Autodesk.Revit.Attributes.Transaction(Autodesk.Revit.Attributes.TransactionMode.Automatic)]
[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("My Dialog Title", "Hello World!");

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

实现外部命令

IExternalCommand 类派生

```
<VB.NET>
''  Hello World #1 - A minimum Revit external command.
<Autodesk.Revit.Attributes.Transaction(Autodesk.Revit.Attributes.TransactionMode.Automatic)> _
<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("My Dialog Title", "Hello World!")

        Return Result.Succeeded

    End Function

End Class
</VB.NET>
```

1.从 IExternalCommand派生新类

实现外部命令

Execute() 方法

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

2.实现 Execute() 方法

```
    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>
```

参数:

1st 访问Revit对象模型

2nd 当命令失败时，返回给用户的消息

3rd A当命令失败时，返回可高亮显示的对象集合

返回值:

- Succeeded
- Failed
- Cancelled

实现外部命令

命令属性

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

        Return Result.Succeeded

    End Function

End Class
</VB.NET>
```

3. 设置命令属性

Transaction mode: 控制命令的的事务模式

- Automatic
- Manual
- ReadOnly

Regeneration option:控制命令中对象修改后的更新模式

- Automatic
- Manual

实现外部命令

显示Hello World

```
<VB.NET>
''  Hello World #1 - A minimum Revit external command.
<Autodesk.Revit.Attributes.Transaction(Autodesk.Revit.Attributes.TransactionMode.Automatic)> _
<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("My Dialog Title", "Hello World!")

        Return Result.Succeeded

    End Function

End Class
</VB.NET>
```

4. 显示一个对话框

任务对话框:
Revit风格的对话框，显示Hello World

用 Visual Studio 开发Revit

- 两种方式来扩展Revit的功能
  - 方式一：外部命令 （External Command）
    - 功能：添加一个Revit命令
    - 描述: 由用户点击按钮来启动命令
    - 使用最频繁
  - 方式二：外部应用（External Application）
    - 功能：做初始化工作
      - 可以添加Ribbon按钮或控件
      - …
    - 描述：在启动和关闭Revit.exe时自动执行
    - 一般会用到，用量不多

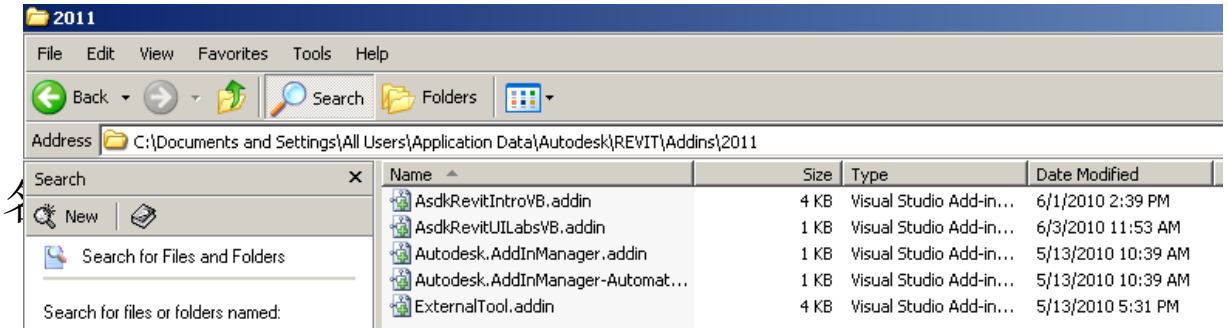
添加外部应用步骤

- 步骤与外部命令相似
- 不同：
  - 新建类从IExternalApplication 接口派生
  - 实现 OnStartup() and OnShutdown() 方法

```
public class App : IExternalApplication
{
  public IExternalApplication.Result
  OnStartup( ControlledApplication application
  {
      CreateRibbonSamplePanel(application);
      CreateRibbonInfosPanel(application);
      return IExternalApplication.Result.Succeeded;
  }
  public IExternalApplication.Result
  OnShutdown( ControlledApplication application )
  {    return IExternalApplication.Result.Succeeded; }
}
```

命令加载

- 方式一: 通过addin 文件
  - *.addin 加载文件
    - 可用任意文件名，只要扩展名
    - 只能存放在指定的目录中



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\

- 在该文件夹下可有多个*.addin 文件并存，Revit依次加载其中设置的加载项目
- 启动Revit后，Revit就加在addin文件中指定的命令

Addin 文件加载清单

.addin File

```
<?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>
```

- Type: Command 或 Application
- Text: 显示在下拉菜单中的字符串
  [插件] tab >> [External Tools] panel
- FullClassName：完整类名，包含命名空间
- Assembly：指向dll的完整路径和文件名
- AddInId：命令的GUID 或独一无二的标示

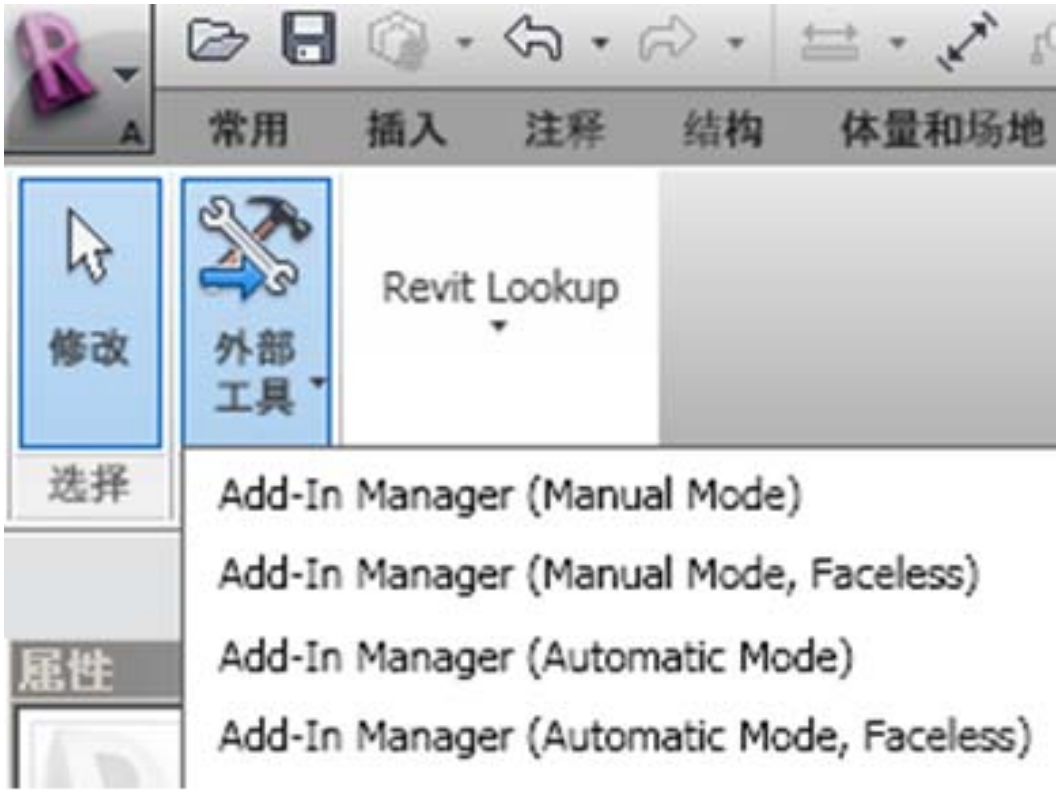
其它更多：

参考 Developer Guide section 3.4.1 (pp40)

命令加载

- 方式二：通过Revit.ini文件
  - Revit.ini 文件
    - 该文件位于产品安装目录的Program子文件夹中
    - 可以加载外部命令和外部应用
    - 在2012中，不再支持这个方法

AddInManager快速加载外部命令



运行命令

运行你的命令

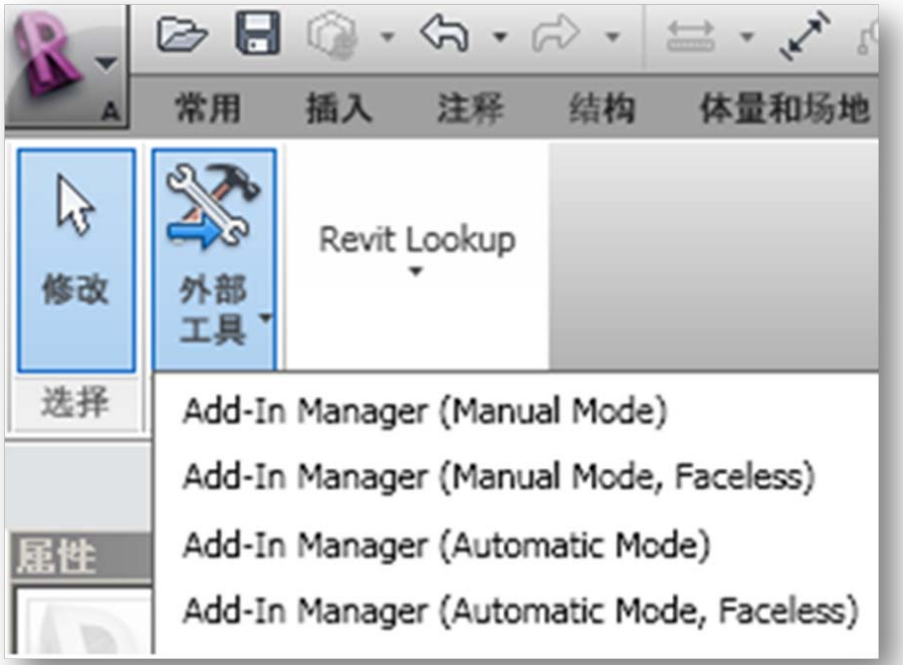
- 当addin加载文件设置好后，启动Revit，在工具栏中有一个“附件模块”，其中的“外部工具”下拉按钮中有加载的命令
- 点击运行加载的外部命令运行

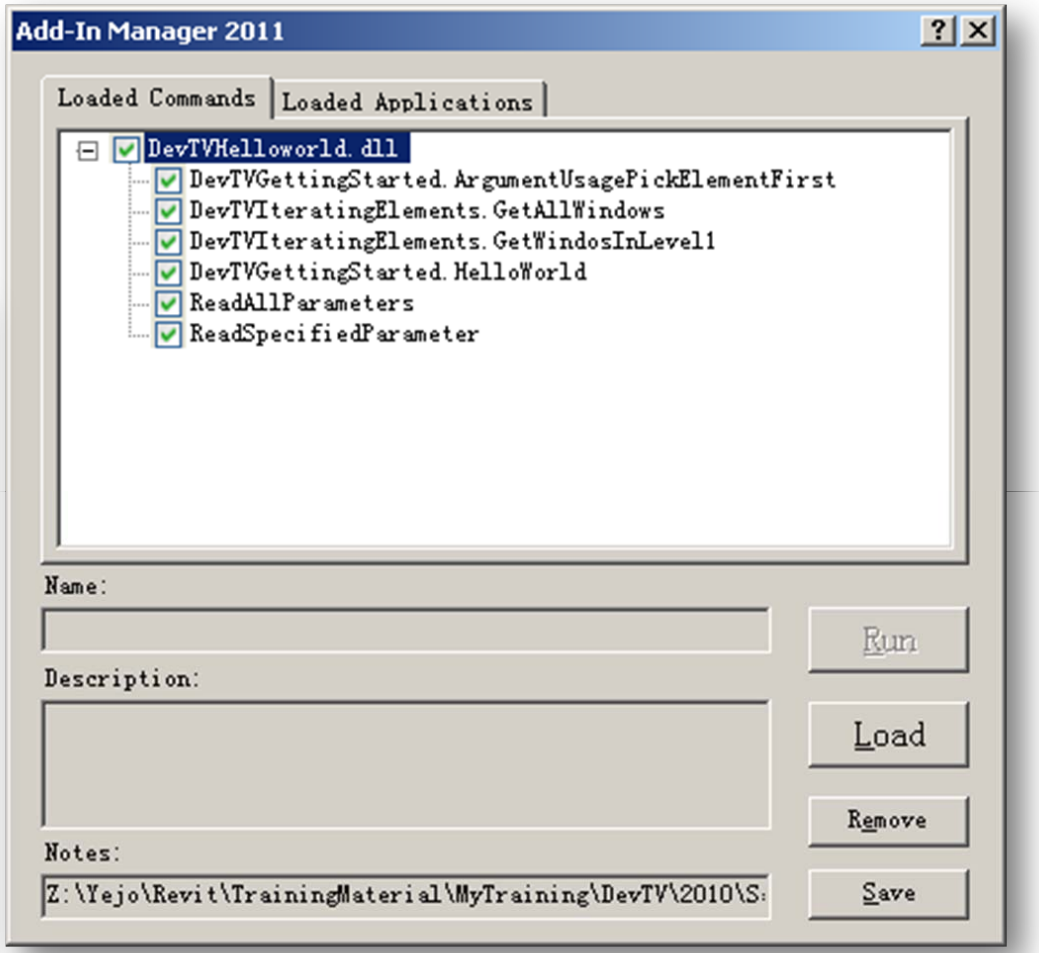
辅助工具

Revit Lookup, Add-In Manager, SDKSamples2011.sln, and RvtSamples

加载工具 AddinManager

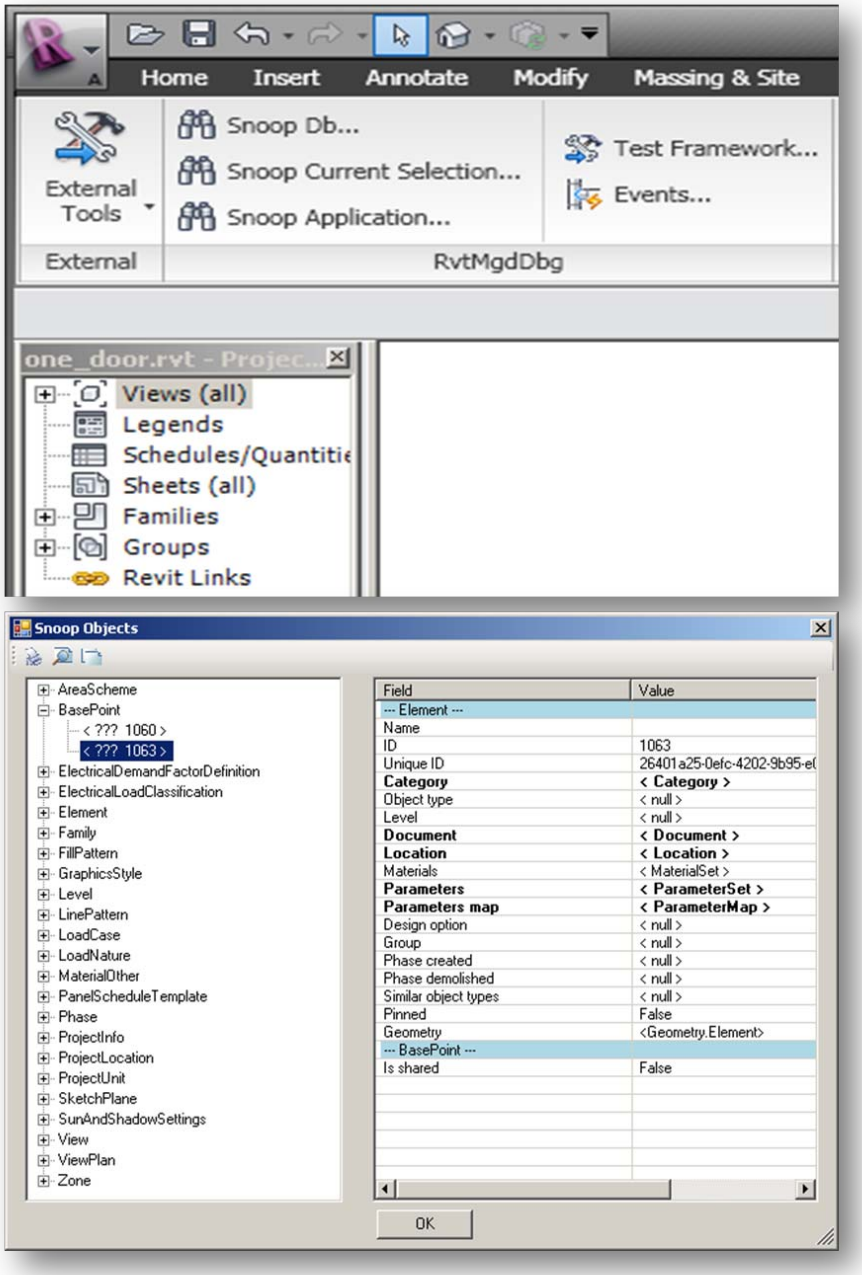
- 加载和卸载用户的程序
- 加载后, 立即可以运行
- 无需手工编辑addin文件
- 自动保存加载命令
- 修改代码编译后再次运行命令
- 包含在SDK中，需预先安装





数据库对象查看工具 RevitLookup

- 源代码在SDK的根目录下
- 读取Revit对象属性和参数值
- 快速查看对象数据



RvtSamples

- 加载所有SDK中的例程命令
- 为每一个命令创建一个命令启动按钮
- 是SDK中的一个开发源代码例程

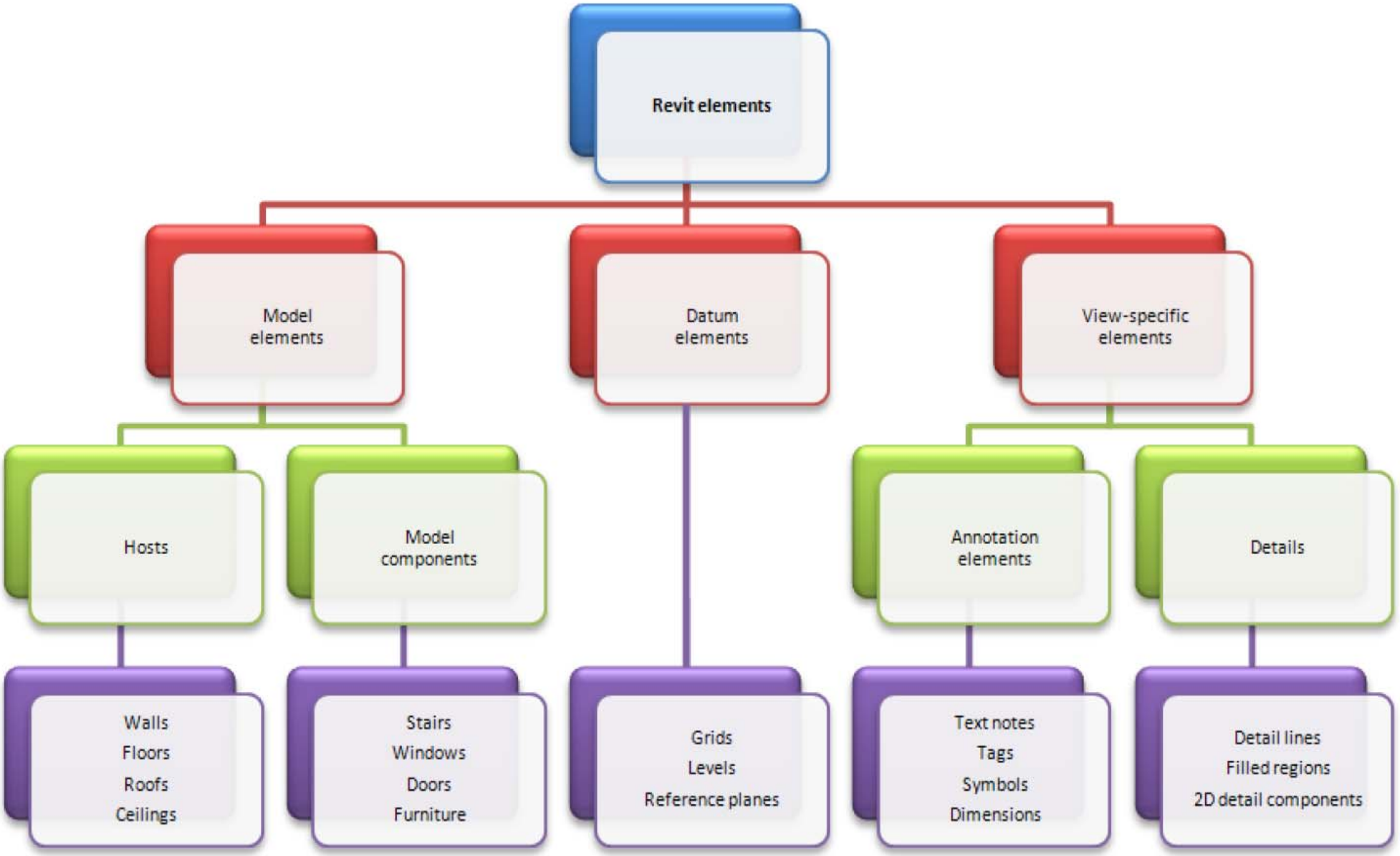
数据库对象

认识Revit 数据库对象

数据库对象

- 数据库对象的API与2010相同
- 详细培训讲解可参考2010的培训录像
  http://download.autodesk.com/media/adn/Revit2010APIWebcast.zip

Revit数据库对象分类



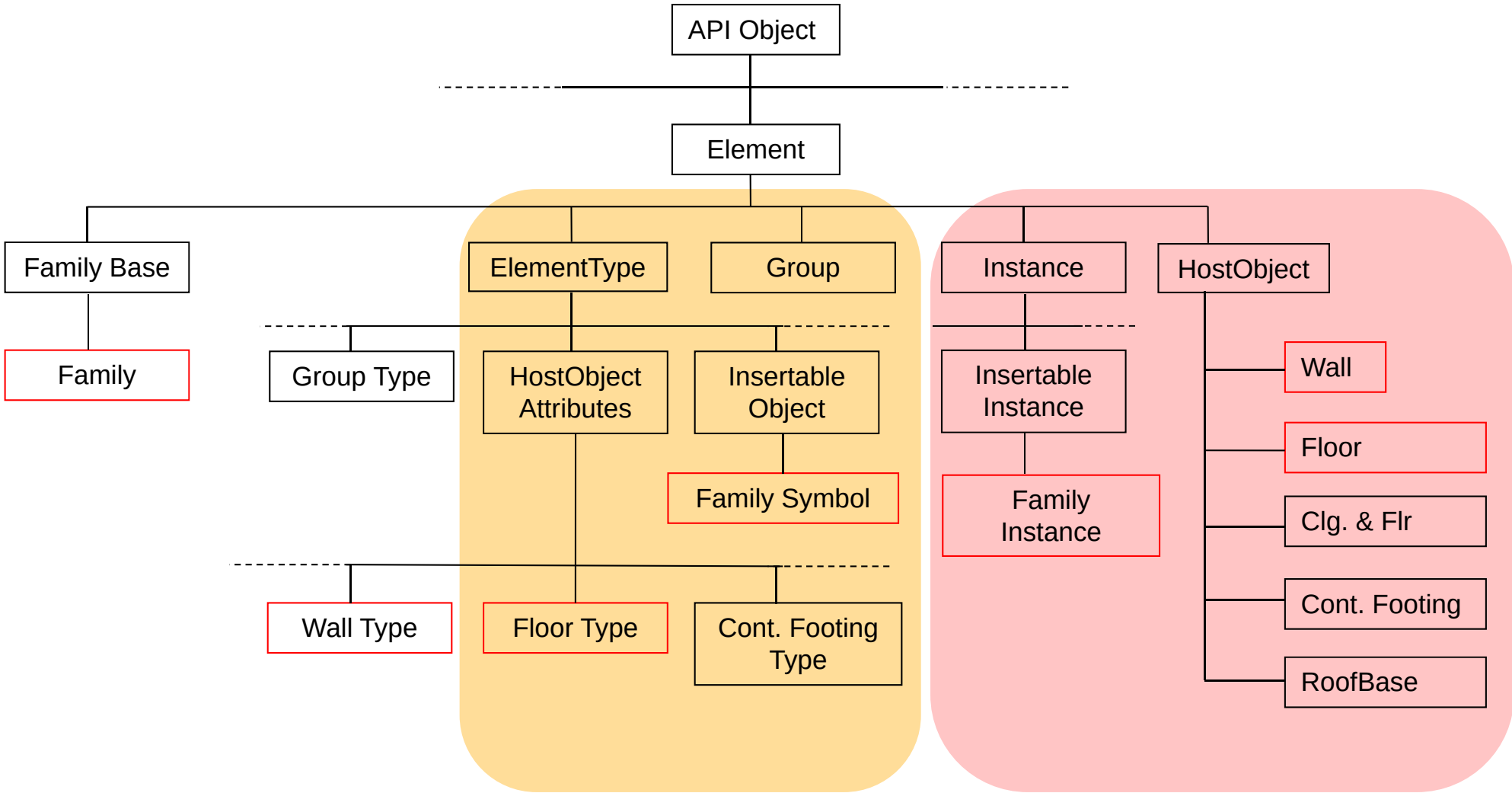
Revit数据库对象的基本特点

- 构件类型，即构件定义
  - 存储在数据库中，作为插入到建筑模型中的对象的模板使用。在建筑模型中不可见。
  - 把相同的数据只保存一份
  - 一个定义，多次使用。
  - 例如: 柱类型

- 构件实例
  - 已经插入到建筑模型中，在视图中可见。
  - 是构件定义的引用，如果构件定义改变，实例跟着改变
  - 例如： 插入到建筑模型中的柱。

数据库对象

类继承关系图



Revit对象的识别

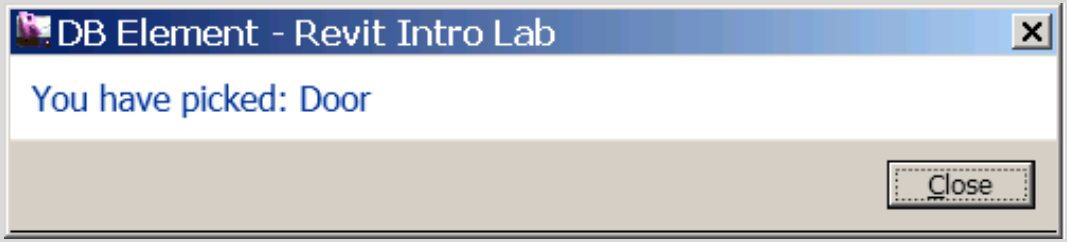
- 对象的类名称
  - 直接可以用类名唯一标识出来。 如系统族: Wall, WallType, Floor, FloorType….
  - 共用一个类. 如标准族：FamilyInstance, FamilySymbol, Family, Element
- 对象的Category 名称.
  - 通过Revit的Category 标识对象的类别
  - 表示同一类别的对象。如：门实例和门类型具有相同的Category
- 其它属性
  - 如所在的楼层
  - 如某一个参数的值

如何识别Revit中不同对象

- 根据标识来识别区分不同对象
- 只使用类名来判断
  - 它们的特点：
    - 片状能容纳其它对象或特定用途的类
    - 如：Wall、Floor、contFooting、CeilingAndFloor等系统族的实例。
- 如果通过类名无法分别出来，需联合对象的类别(Category)来判断
  - 门、窗、柱等对象没有专用的类来表示，都是FamilyInstance的实例
  - 用Category来判断其类别
    - 用枚举型的BuiltInCategory, 来创建ElementCategoryFilter对象, 支持所有国家语言。

DB Element

识别对象




```
<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>
```

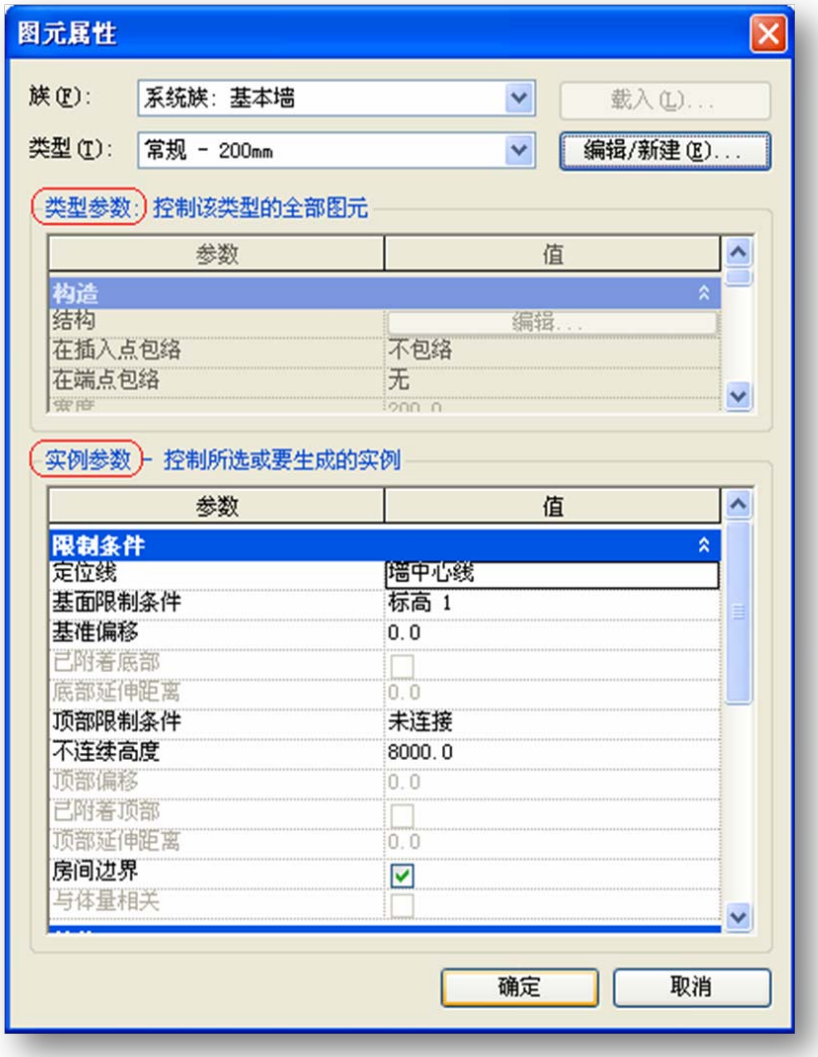
参数

作用：可保存对象的任何属性

分类：

内置参数和共享参数

实例参数和类型参数



访问实例所有参数

Element.Parameters 属性

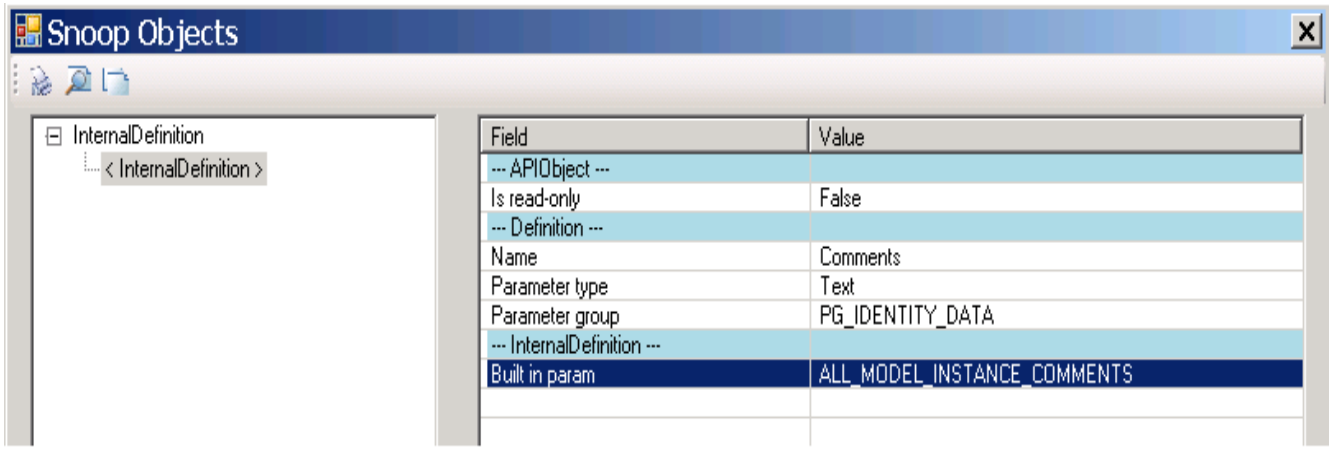
VB.NET>

```
    ''  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>

访问指定参数

- 四种方法获取特定参数
  - Parameter(parameterId As Parameters.BuiltInParameter)
  - Parameter(definition As Parameters.Definition)
  - Parameter(guid As System.Guid)
  - Parameter(name As String)



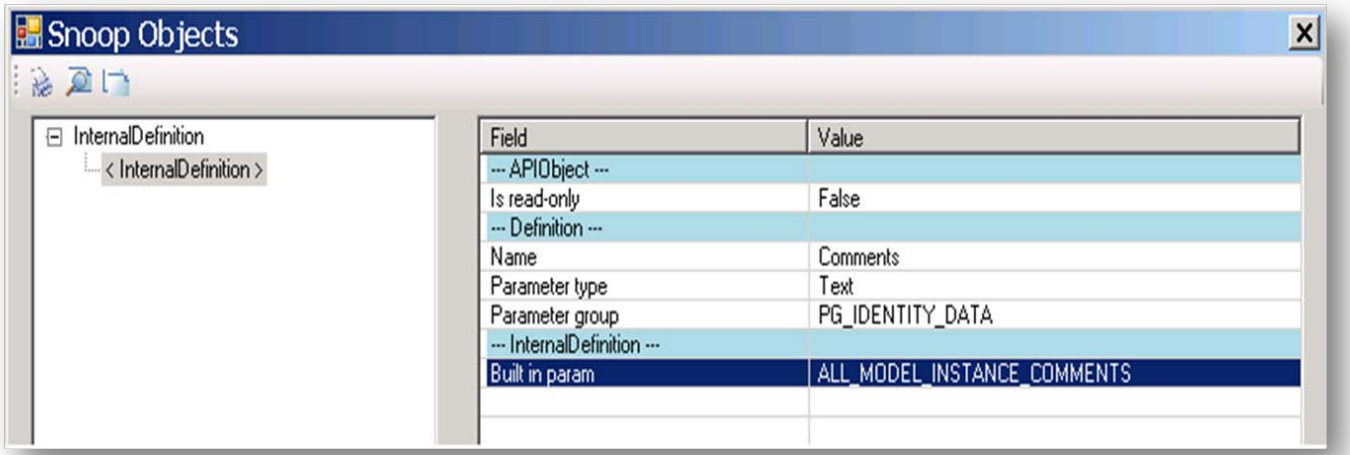
内置参数功用 **BuiltInParameter**

- 与国家语言无关
- 有些参数无法直接用参数名称字符串来访问，但是可以用内置参数来访问

  param = elem.Parameter(

  BuiltInParameter.SYMBOL_FAMILY_NAME_PARAM)

- RevitLookup快速查参数的内置参数名



参数和内置参数

```
<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("Mark")
        ...
        ''  (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>
```

数据库对象的位置

- Element.Location 属性
- 有两种Location对象
  - LocationPoint – 基于点的位置 (e.g., 家具)
  - LocationCurve – 基于线的位置 (e.g., 墙)
- 需要将返回的Location对象转成 LocationPoint 或 LocationCurve

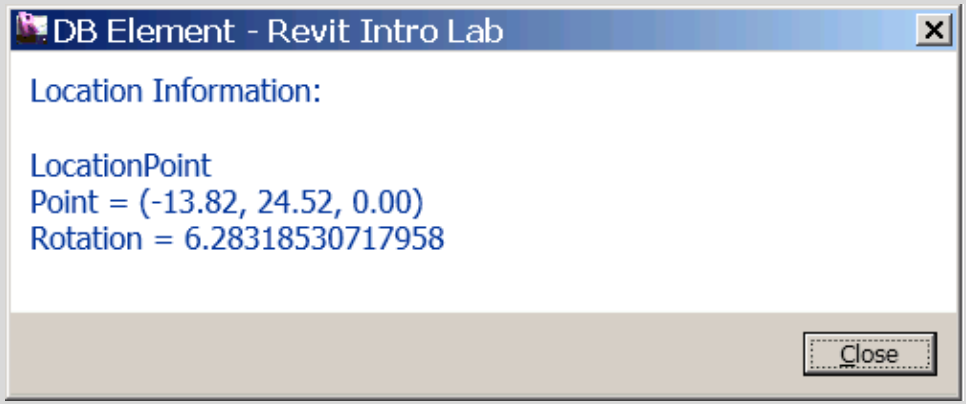
DB Element

Location

<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>



几何信息和数据

- Geometry Options – 指定返回什么样的几何信息，如粗略，详细
- 几何对象有下面这几类对象
  - Solid
  - Geometry Instance
  - Curve
  - Mesh.
- Solid可以再细分成Solids/Faces/Edges – 可用RevitLookup查看
- SDK中的例子包含了如何访问对象的几何数据
  - ElementViewer
  - RoomViewer
  - AnalyticalViewer

对象过滤

2011 重要的功能优化

对象遍历，过滤和查询

对象过滤

获取一个对象

- 在Revit中所有对象都放在一起，好比都一个麻袋中
- 为了获得感兴趣的对象，需要遍历，过滤或查询这个麻袋
- 典型的情况我们可能会:
  1. 获取族的所有类型 (e.g., 墙类型，门类型)
  2. 获取一个指定类的实例
     (e.g., 所有墙对象, 所有门实例)
  3. 找到一个指定名称的族类型
     (e.g., “Basic Wall: Generic – 200mm”, “M_Single-Flush: 0915 x 2134mm”)
  4. …

对象过滤

1.1 获取所有族类型 – 系统族

- 获取所有墙类型
- 三种方法(后两种方法使用了Of***快捷过滤)

```
<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>
```

对象遍历

1.1获取指定族类型 – 系统族

- 获取墙类型e.g., “Basic Wall: Generic – 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>
```

对象遍历

1.2 获取指定族的所有类型 – 标准族

- 获取门的所有类型，两种方式均可

```
<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 doorTypeCollector = New FilteredElementCollector(m_rvtDoc)
        ElementClassFilter classFilter = new ElementClassFilter(GetType(FamilySymbol))
        ElementCategoryFilter catFilter = new
                              ElementCategoryfilter(BuiltInCategory.OST_Doors)
        LogicalAndFilter logFilter = New LogicalAndFilter(classFilter,catFilter)
        doorTypeCollector.WherePasses(logFilter)
        Dim doorTypes As IList(Of Element) = doorTypeCollector.ToElements
</VB.NET>
```

对象遍历

1.2 获取指定类型 – 标准族

- 获取门类型 e.g., “M_Single-Flush: 0915 x 2134”

```
<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>
```

对象过滤

2.1 获取指定类的实例 – 系统族

- 获得所有墙

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

对象过滤

2.1 获取指定类的实例 – 系统族

- 例如, 获取门，其类型是 “M_Single-Flush: 0915 x 2134”

<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>

对象过滤

2.2 获取指定类的实例 – 标准族

- 获取所有门

```
<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>
```

对象过滤

更多选择

我们已经演示了如下类的使用:

- FilteredElementCollector
- ElementClassFilter
- ElemetCategoryFilter

有更多的过滤器可供使用:

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

FilteredElementCollector 用法

- 用快捷函数添加过滤条件
  - OfCategory
  - OfCategoryId
  - OfClass

- 过滤器
  - 使用单一过滤器
  - 多个过滤器的逻辑组合过滤器

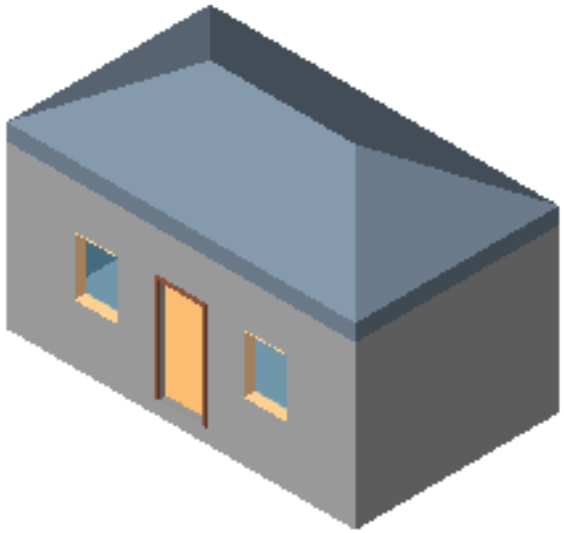
- 使用LINQ
  - .NET 3.5 提供功能
  - 可以对集合中的对象查询

对象修改

如何修改一个对象， 与Revit2010 相同

编辑Revit对象

- 修改对象的方式
  - 通过类的成员函数或属性直接修改
    - 修改对象的类型 Door.Symbol = newSymbol
  - 修改对象的参数值.
    - Parameter.Set()
  - 修改模型构件的几何位置
  - 通过Document提供的方法修改
    - Document.Rotate()
- 删除
  - Document.Delete(…)方法



修改对象

- 修改族类型 (例如, 墙、门的类型)

```
<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), "Basic Wall", "Exterior - Brick on CMU")
    ''  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), "M_Single-Flush", "0762 x 2032mm", _
         BuiltInCategory.OST_Doors)
    ''  assign a new family type.
    aDoor.Symbol = newDoorType
</VB.NET>
```

对象修改

- 修改对象的参数值

```
<VB.NET>
aWall.Parameter(BuiltInParameter.WALL_TOP_OFFSET).Set(14.0)
aWall.Parameter(BuiltInParameter.ALL_MODEL_INSTANCE_COMMENTS).Set( _
     "Modified by API")
</VB.NET>
```

对象修改

- 修改对象的位置

```
<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>
```

对象修改

Document级别的移动，旋转等

- 移动，旋转对象

```
<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>
```

对象修改

更新对象

- 修改完一个对象后，如果想在紧接着的代码操作是在修改后的对象上进行，那么需要更新所作的所有修改
- 如果你的RegenerationOption 是Manual， 需要调用

```
m_rvtDoc.Regenerate()
```

- 如果RegenerationOption 是Automatic ，每次修改后，Revit自动更新对象到最新状态

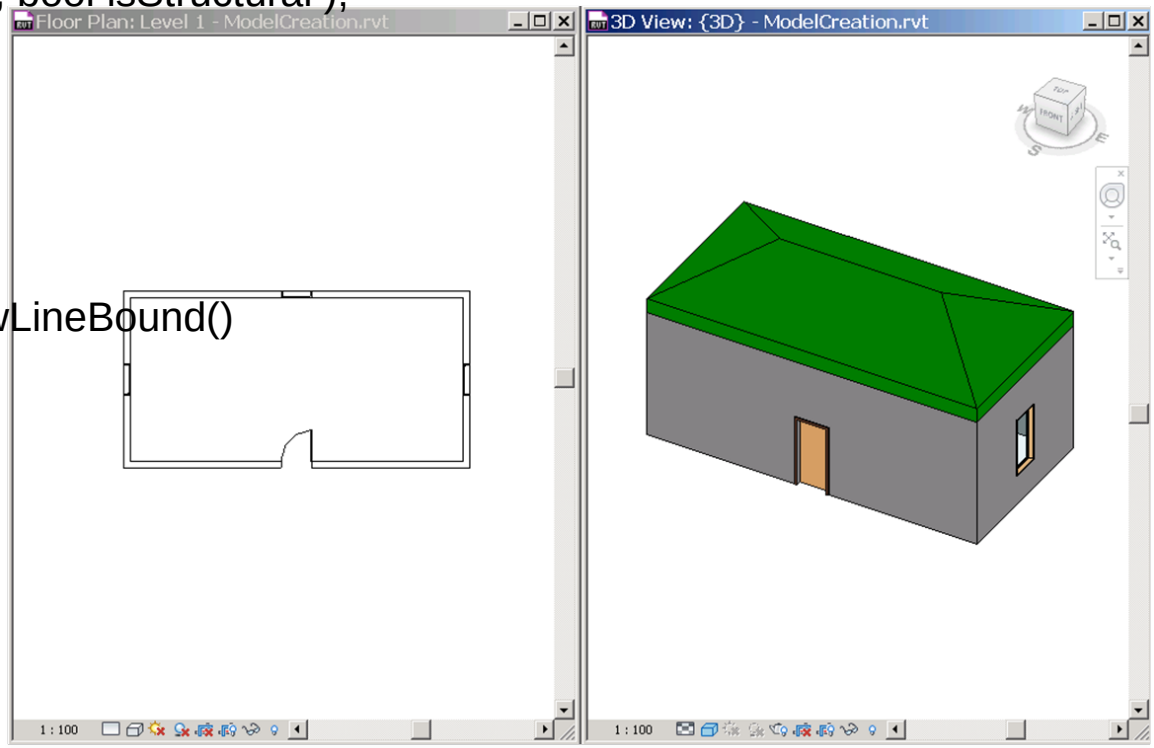
创建对象

如何创建Revit中的各种对象

创建对象

- 创建模型对象
  - 在类Autodesk.Revit.Creation.Document中定义了120 个对象创建方法
  - 支持创建40多种类型的对象
  - 例如 Walls, Floors
    - NewWall(CurveArray profile, bool structural); // + 4 个重载函数
    - NewSlab(CurveArray profile, Level, Line slopedArrow, double angle, bool isImperial, bool isStructural);
    - NewFamilyInstance（…..)

- 创建临时对象
  - 这类对象不保存在模型文件中
  - Application.Create.NewXxx() 如, NewLineBound()



模型创建

新建墙

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

模型创建

新建门

```
<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), _
          "M_Single-Flush", "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 API Intro Labs 介绍

- Revit API 基础部分内容
  - Revit 插件: 外部命令/应用, 属性, 插件加载和对象模型
  - Revit对象的特点
  - 对象遍历，过滤和查询
  - 对象修改
  - 模型创建
- Exercises:
  - Lab1 – “Hello World”
  - Lab2 – DB element
  - Lab3 – element filtering
  - Lab4 – element modification
  - Lab5 – model creation

休息5分钟…

Revit 界面交互编程

Ribbon, 选择集操作，任务对话框，事件，模型动态更新

Ribbon API

如何修改用户交互界面

Ribbon API

概览

- Ribbon API 是唯一修改用户交互界面的手段
  - 原来的菜单和工具条需要移植到Ribbon
- 容易使用
- 无需WPF知识
- 向导帮助文件
  - Ribbon design guidelines.pdf
  - Autodesk Icon Guidelines.pdf

Ribbon API

- API创建的控件只能位于 “附加模块” 页或 “结构” 页
- 用addin文件加载的命令都位于“外部工具”下拉按钮下
- 在外部应用中创建Ribbon的各种控件
  - 按钮
  - 下拉按钮
  - 独立或层叠布局的按钮（二行或三行）
  - 分隔按钮 *(new!)*
  - 单选按钮组 *(new!)*
  - 组合框 *(new!)*
  - 文本框 *(new!)*
  - 向下滑动扩展区 *(new!)*

Ribbon API

相关类

- RibbonPanel
  - 代表Ribbon 中的一个方框
  - 其中可以放一些按钮或其它控件
- RibbonItem
  - 按钮，下拉按钮，组合框，文本框，单选按钮的基类
- PushButton, PushButtonData
  - 按钮信息
- PulldownButton, PulldownButtonData
  - 下来按钮信息
- SplitButton, SplitButtonData
  - 分隔按钮信息
- ComboBox, ComboBoxData
  - 组合按钮
- …

Ribbon API

2011新功能

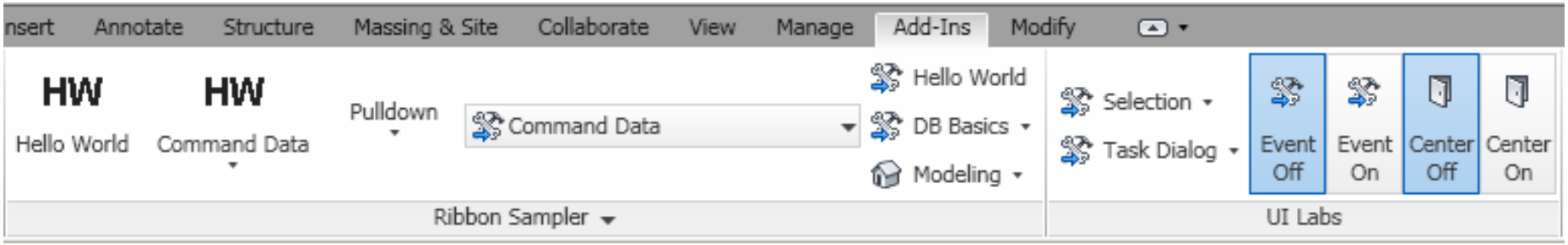
- 新的命名空间

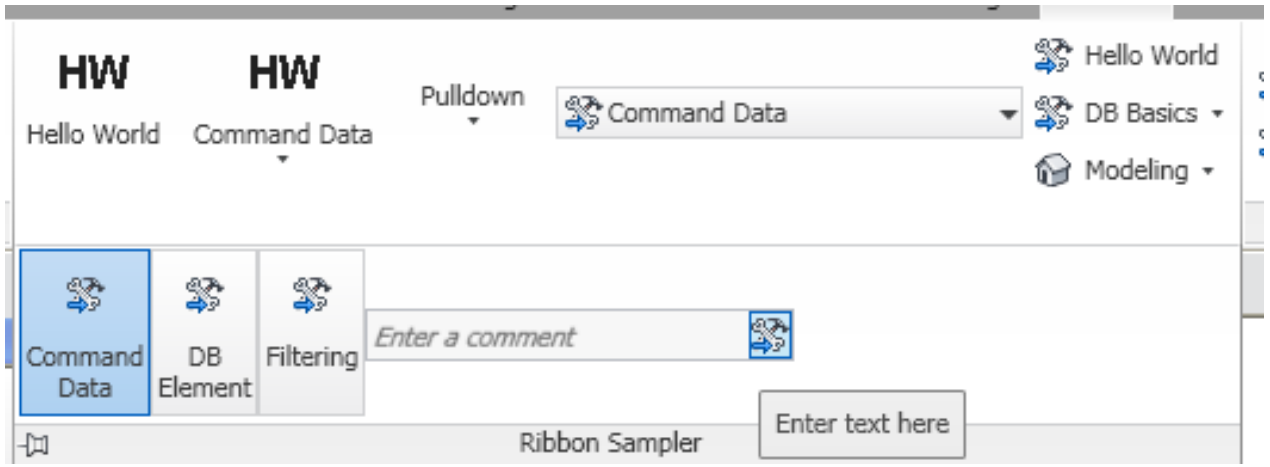
  Autodesk.Revit.UI

- 新的界面控件 (SplitButton, ComboBox, TextBox, etc)
  - ComboBox 和 TextBox新的事件

- 新的属性
  - RibbonItem.Visible
  - RibbonItem.LongDescription
  - RibbonItem.ToolTipImage
  - PushButton.AvailabilityClassName

Lab - Ribbon API





交互选择

在用户程序中选择点，选择对象

交互选择

概要说明

- 能选择对象，选择点，对象边界和对象的面
- 将对象加入到当前选择集:
  - PickObject()
  - PickObjects()
  - PickElementsByRectangle()
  - PickPoint()

```
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);
}
```

交互选择

对象捕捉

- 在选择点时，可以设置捕捉类型

```
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);
}
```

- 可以设置工作平面
  - View.SketchPlane()

交互选择

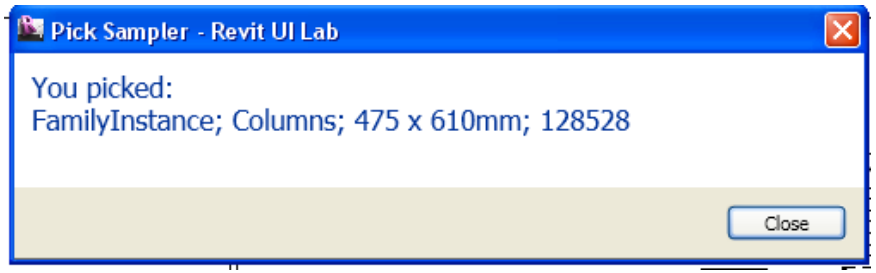
选择过滤器

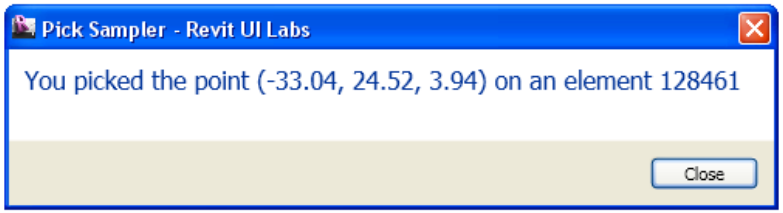
- ***ISelectionFilter*** *接口用来设置选择时对象过滤*
  - AllowElement()
  - AllowReference()

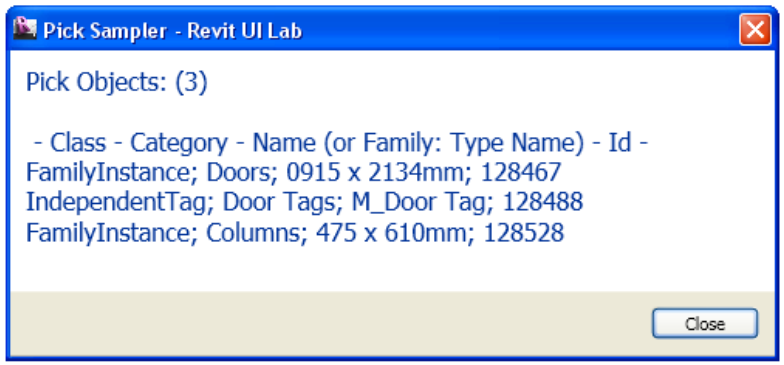
```
public void SelectPlanarFaces(Autodesk.Revit.DB.Document document)
{
		UIDocument uidoc = new UIDocument(document);
		ISelectionFilter selFilter = new PlanarFacesSelectionFilter();
		IList<Reference> faces = uidoc.Selection.PickObjects(ObjectType.Face, selFilter, "Select multiple planar faces");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

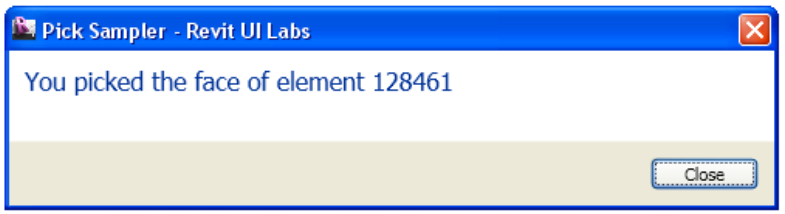
演示 – 交互选择

选择的例子



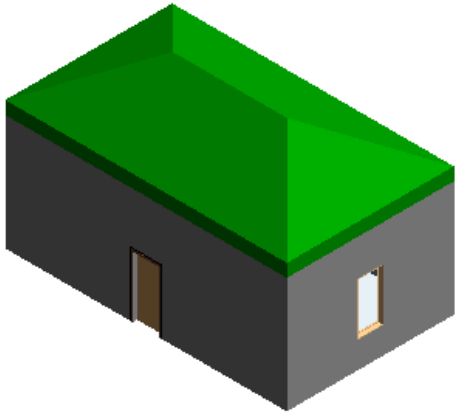






交互选择

创建一个房子，选择点来定位房子的对角位置



```
<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>
```

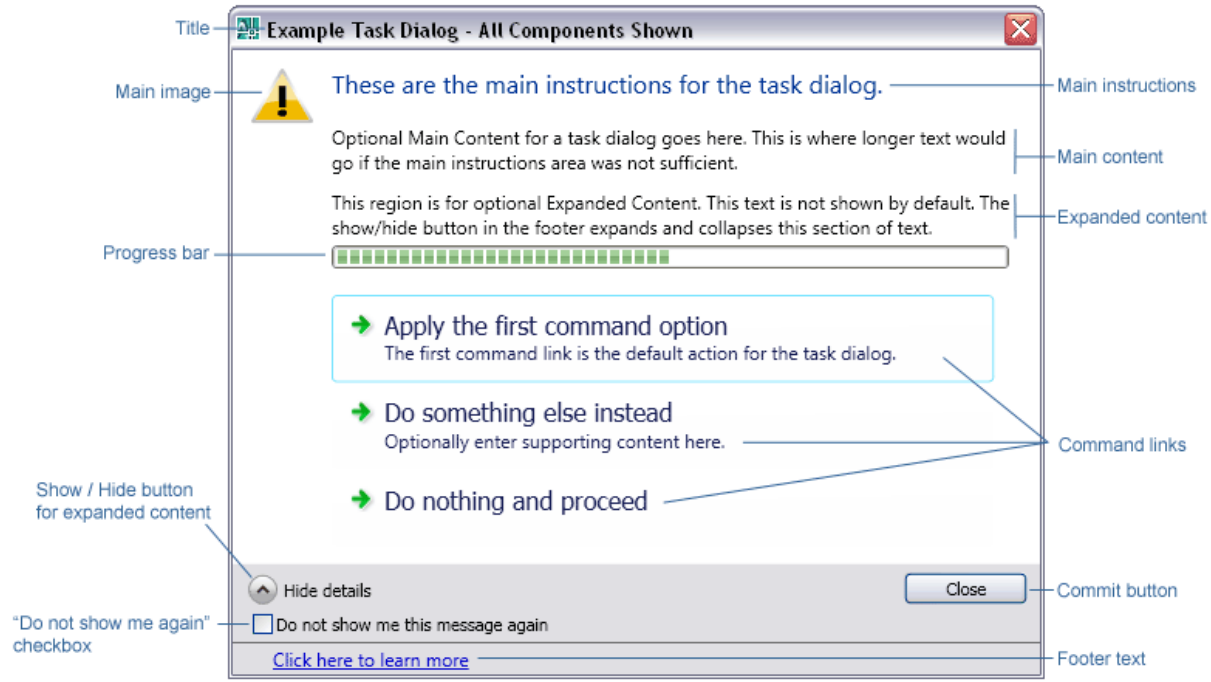
任务对话框

Revit 风格的消息框

任务对话框

概览

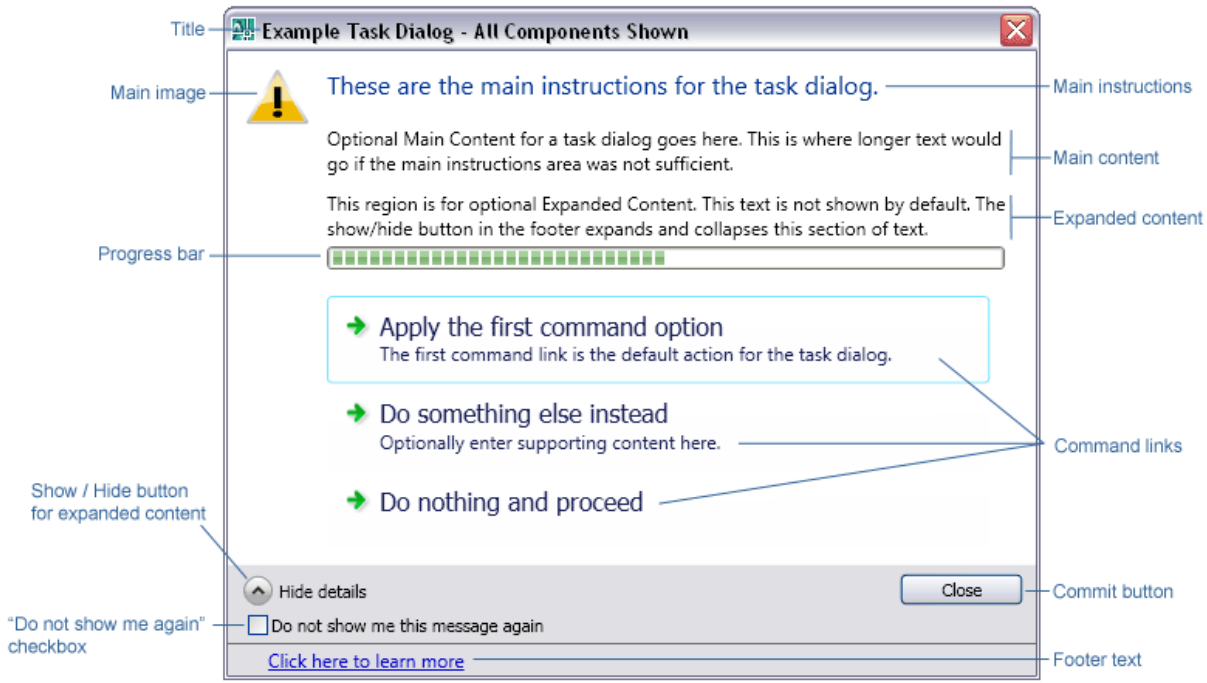
- 模式对话框，上面有系列控件
- Revit 风格，不是普通的Windows 消息框
- 当需要时，可以用
  - 提供信息
  - 问一个问题
  - 允许用户定制需要的控件项目



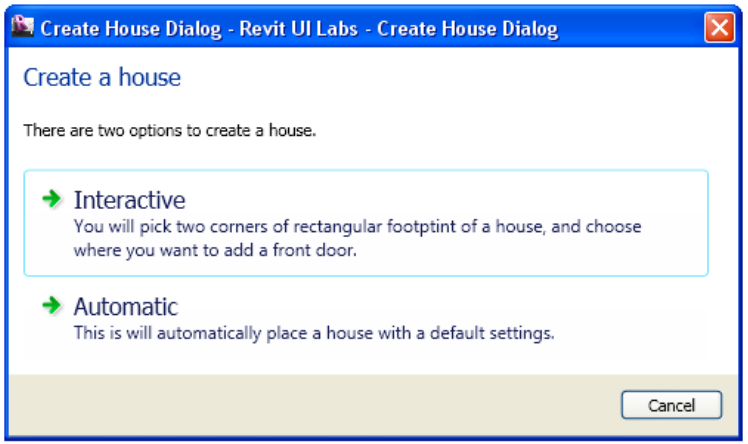
任务对话框

- 两种方式创建任务对话框：
  - 构件一个 TaskDialog对象, 设置属性并使用实例的Show（）方法来显示
    - Instance of Autodesk.Revit.UI.**TaskDialog**
  - 使用一个 static Show() 一下子显示一个任务对话框

- 可设置的界面元素包括：
  - 概要性文字
  - 详细说明文字
  - 图标
  - 按钮
  - 命令链接
  - 确认文本等



任务对话框例子




```
<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>
```

事件和模型动态更新

应用程序级，文档级，以及对象级的事件

事件

- 当特定操作/事情发生时，触发通知
- 与.NET的事件用法一致
  - Pre and Post 事件
  - 单一事件 (DocumentChanged and FailureProcessing)
- VSTA目前不支持事件

- 类型：
  - 应用程序级别事件
  - 文档级别事件
  - 对象级别的事件

事件

- 在事件处理函数中判断是否可以修改Document
  - Document.IsModifiable
  - Document.IsReadOnly

- 多数Pre-events可以取消，如DocumentSaving 事件可取消
  - RevitEventArgs.Cancellable
  - RevitAPIPreEventArgs.Cancel

事件

事件响应函数，事件注册，取消事件注册

- 事件处理函数EventHandler

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

}
```

- 事件注册

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

- 取消事件注册

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

事件

2011新功能

- 命令空间的调整
  - Autodesk.Revit.DB.Events
  - Autodesk.Revit.UI.Events

- 新事件
  - Document.Changed ： 当文档中的对象发生增删改时触发
  - UIApplication.Idling ： 当Revit处于空闲时触发

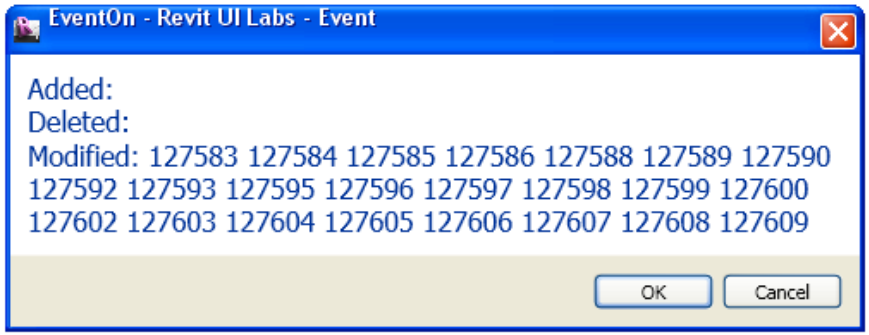
事务

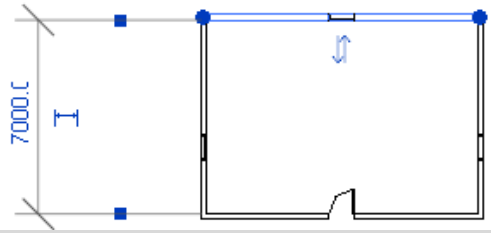
2011新功能

- 在事件处理函数中不再自动打开一个事务。

- 下面这些2010之前的事件去掉
  - Application.OnDocumentSaved
  - Application.OnDocumentSavedAs
  - Application.OnDocumentOpened
  - Application.OnDocumentClosed
  - Application.OnDocumentNewed
  - Application.OnDialogBox
  - Document.OnSaveAs
  - Document.OnSave
  - Document.OnClose

例子





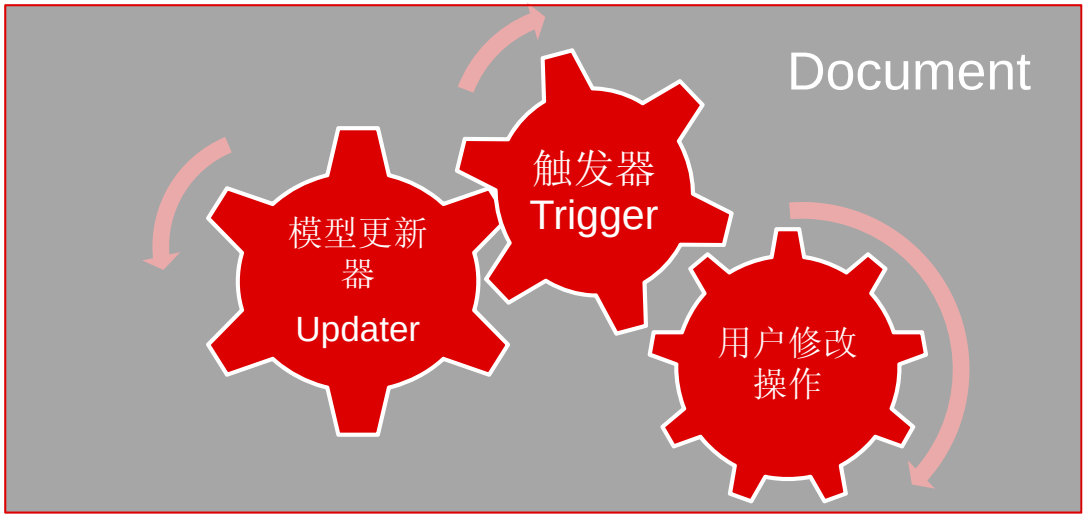
```
//' register the document changed event
application.ControlledApplication.DocumentChanged += UILabs_DocumentChanged;
```

```
//' 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 = "Added: ";
foreach (ElementId id in idsAdded)
{
   msg += id.IntegerValue.ToString() + " ";
 }
```

模型动态更新

概览

- Dynamic Model Update
- 功能：当用户作出一个操作，API程序能捕获该动作，并且对模型同时做修改
- 能捕获添加新对象，修改对象，以及对象删除这些操作



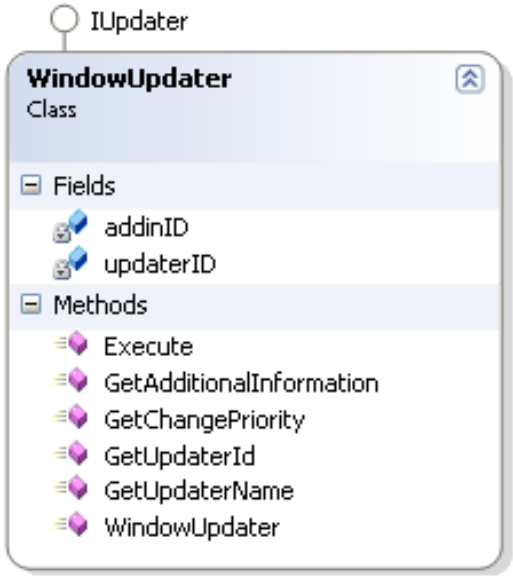
模型动态更新

Updaters

IUpdater 功能:

- 提供方法来修改模型。当指定操作发生时，触发这个方法。

- *IUpdater* 如下的方法.
  - GetUpdaterId ()
  - GetUpdaterName()
  - GetAdditionalInformation ()
  - GetChangePriority()
  - Execute()



模型动态更新

注册和触发

- 注册Updater
  - 在OnStartUp 注册应用程序范围的更新器
  - 在ExternalCommand 中注册命令级的更新器

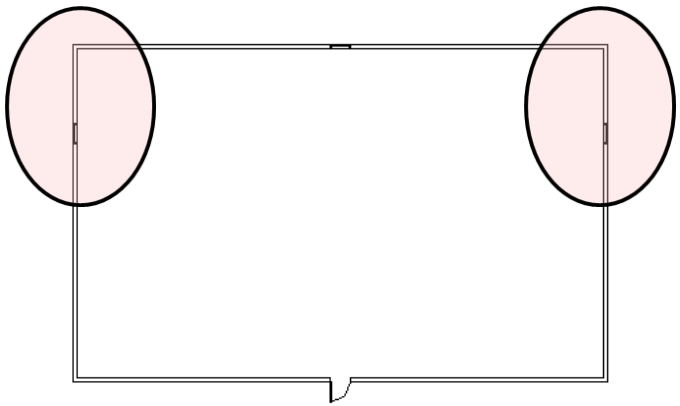
```
WindowUpdater updater = new WindowUpdater(application.ActiveAddInId );
// Register the updater in the singleton UpdateRegistry class
UpdaterRegistry.RegisterUpdater( updater );
```

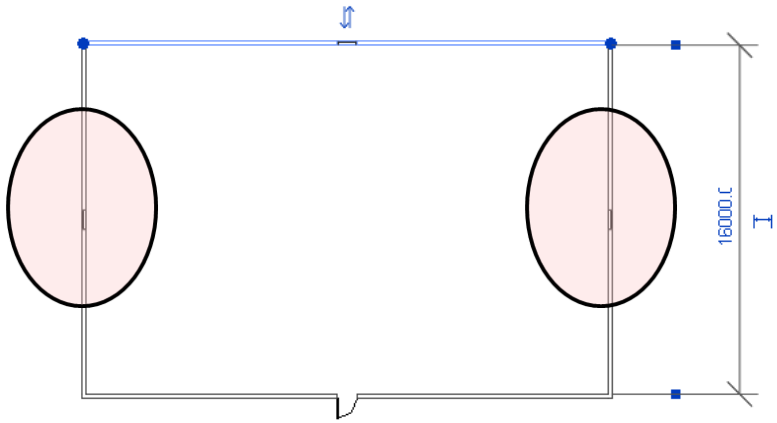
- 添加触发器
  - 对象范围： ElementId 集合或用ElementFilter表达的对象集合
  - 操作类型： 对象添加，删除，修改

```
// 用过滤器来表达对象范围
ElementClassFilter filter = new ElementClassFilter( typeof( Wall ) );
// 添加触发器
UpdaterRegistry.AddTrigger(updater.GetUpdaterId(),filter,
Element.GetChangeTypeGeometry());
```

模型动态更新例子——窗户始终居中

```
//' 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());
```





尾声

Where do we go next …

今天讲的内容

- Revit二次开发起步
  - 产品和SDK介绍
  - Revit 二次开发环境
  - Revit 二次开发的完整过程
- Revit数据库对象
  - 数据库对象特征
  - 过滤数据库中对象
  - 访问和编辑对象的参数
  - 几何数据访问
  - 操作Revit数据库中对象
- Revit 2011 API 新功能
  - Ribbon 界面编程
  - 选择集应用
  - 任务对话框
  - 事件和模型动态更新

培训反馈调查

深入学习

SDK中的资源

Revit 2011 API DevTV中文

- http://download.autodesk.com/media/adn/Revit_2011_API_DevTV_Chinese.zip

- 论坛讨论组
  - http://discussion.autodesk.com > Revit Architecture > Revit API

API 培训课程

- http://www.autodesk.com/apitraining

- The Building Coder, Jeremy Tammik 的 Revit API 博客
  - http://thebuildingcoder.typepad.com

加入ADN

- http://www.autodesk.com/joinadn

- ADN网站有大量常见问题解决方案（只有ADN用户才能访问）
  - http://adn.autodesk.com

AEC 开发者夏令营

六月 7-9, 2010 Sheraton Boston, 马塞诸萨州
面向开发和定制的 3 天培训营会
任何人对基于Autodesk建筑行业的产品开发定制感兴趣的人都可以参加
包含 “.NET programming for non-programmers”

很好的机会来实现:

- 学习Autodesk最新的技术
- 和Autodesk管理层和工程师交流
- 可以有一对一的交流
- 与其它客户或开发者交流

更多信息请看:
http://devcamps2010.autodeskevents.net/

Revit API 培训和免费现场服务

Revit 二次开发DevLabs （二天）

- Aug 9~ 10　　北京, Autodesk的会议室（国贸大厦2座7层）

- AutoCAD 二次开发DevLabs （三天）
  - Aug 4~6　　北京, Autodesk的会议室（国贸大厦2座7层）

Revit API 教室培训

- July 22 & 23　　北京

如果想参加，请在网站登记注册:

Developer Center >> API Training and Consulting >> Schedule

http://www.adskconsulting.com/adn/cs/api_course_sched.php

今日培训实况可下载...

今日培训过程被录制，并将发布到网上:

ADN 网站

http://adn.autodesk.com
Revit > Knowledgebase > Whitepapers and Training Videos

官网开发者中心（Developer Center）

API Training & Consulting > Schedule
http://www.autodesk.com/developer

反馈调查

谢谢~~

Questions & Answers

Autodesk®