

Lab1 – Hello World

Created by M. Harada, July 2010

Updated by DevTech AEC WG

Last modified: 6/9/2017

<C#>C#バージョン</C#>

目的:この実習では、Revit API の基本を学習します。学習する項目は、次のとおりです。

- 外部コマンドあるいはコマンド・レベル・アドインを実装する
- 外部アプリケーションあるいはアプリケーションレベル・アドインを実装する
- 外部コマンド、または、外部アプリケーション定義に属性を適用する
- 外部コマンドや外部アプリケーションを Revit に登録するためにアドイン・マニフェスト・ファイルを記述する
- 外部コマンド・データによって Revit モデルにアクセスする

さらに、次のツールを習得する機会を得ることができます：

- RevitLookup
- アドイン・マネージャ

タスク:この実習では、2 つのコマンドと 1 つのアプリケーションを記述します：

1. **“Hello World”** — 「Hello World」メッセージをダイアログに表示させる、最小の外部コマンド。このコマンドは、[アドイン]タブ>>[外部ツール]パネルから実行します。
2. **“Hello World App”** — 「Hello World from App」メッセージをダイアログに表示させる、最小の外部アプリケーション。ダイアログは Revit の起動時に自動的に表示されます。
3. **“Command Data”** — ExternalCommandData あるいは IExternalCommand.Execute() メソッドの最初の引数を参照し、そこから取得した情報を記述する外部コマンド。

図 1～3 は、実習で定義するコマンドとアプリケーションの起動からのサンプル画像を示します：

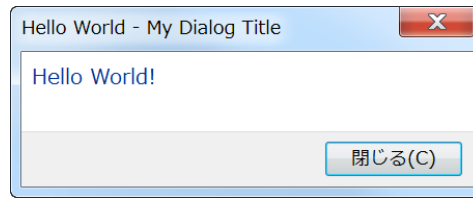


図 1.「Hello World」コマンド

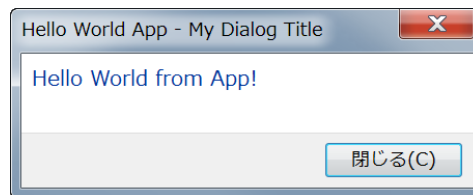
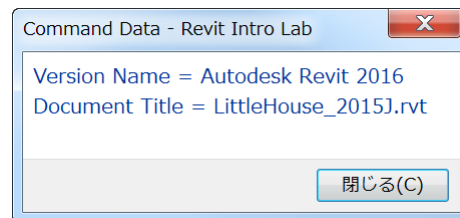


図 2.「Hello World from App」アプリケーション



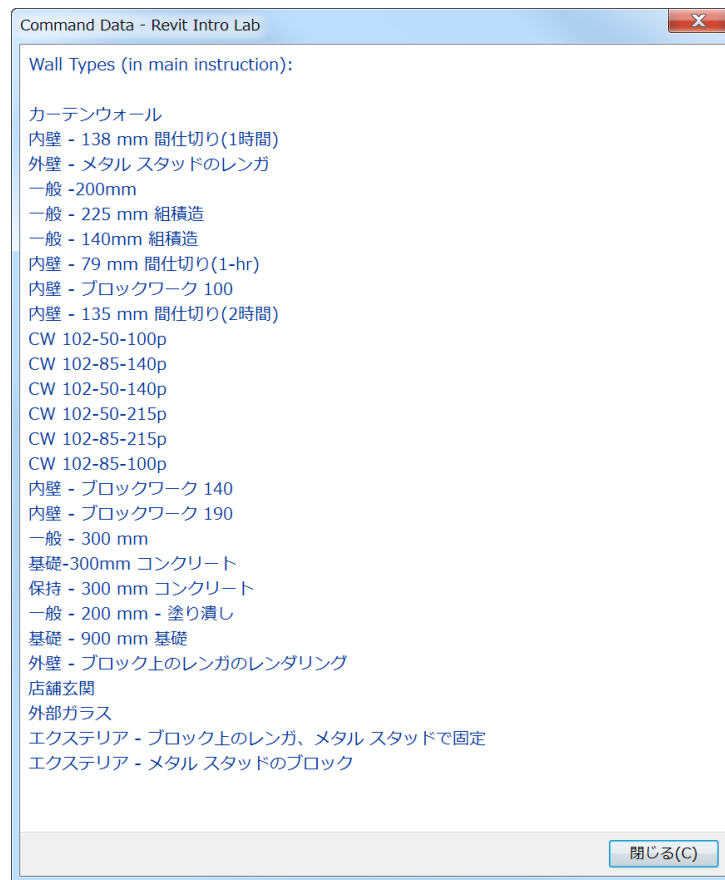


図 3.「CommandData」コマンドは、ExternalCommandData の情報を表示

下記は、この実習の実装と確認の手順です:

1. 「Hello World」: 外部コマンド
2. アドイン・マニフェスト・ファイル
3. コマンドのテスト
4. 名前空間の完全修飾名を省略して「Hello World」
5. 「Hello World App」: 外部アプリケーション。
6. CommandData および Revit オブジェクト・モデル

1. 「Hello World」:外部コマンド

Revit アドインは、Microsoft Visual Studio のクラス ライブラリとして定義されます。

Microsoft Visual Studio に慣れていない場合は、Revit API 開発者用ガイド の「[Walkthrough: Hello World](#)」項目を参考にしてください。VB.NET を使用する場合は、「[VB.NET の Hello World](#)」を参照してください。

1.1 好きな開発言語(VB.NET または C#)のクラス ライブラリ・テンプレートを使用して、新しい Visual Studio プロジェクトを作成します。ここでは C#を選択して以下のように指定します:

- ソリューション名: **IntroCs**
- プロジェクト名: **IntroCs**
- ファイル名: **1_HelloWorld.cs**
- コマンドクラス名:**HelloWorld**(この後、定義します)

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

要求されるアセンブリ参照:

プロジェクトに参照を加えてください。少なくとも、次の参照が必要となります:

- System.dll
- System.Core.dll(LINQ 照会用)
- RevitAPI.dll (Revit のインストール フォルダから)
- RevitAPIUI.dll (Revit のインストール フォルダから)

アセンブリを参照する場合には、参照プロパティで「ローカル コピー」を False に設定してください。

要求される名前空間:

デフォルト設定に加えて、次の名前空間をプロジェクトに加えてください:

- System.Linq
- Autodesk.Revit.DB
- Autodesk.Revit.UI
- Autodesk.Revit.Attributes
- Autodesk.Revit.ApplicationServices(Revit アプリケーション用)

注意:VB.NET と C# の振る舞いの違いについて

VB.NET を使用している場合、上記の名前空間を次の UI から設定することができます: [プロジェクト]>>[プロパティ]>>[参照]>>[参照された名前空間]。実際、各.vb ファイルの中で「Import」キーワードを使用する代わりに VB.NET プロジェクトのプロジェクト プロパティの中で名前空間をセットすることを推奨します。そうでない場合には、次のようなエラーメッセージに遭遇する可能性があります:

'XXX' は 'Friend' であるため、このコンテキストではアクセスできません。

プロジェクト・レベルでインポート設定を使用すれば、このエラーは回避することができます。

C# では、プロジェクト・レベルに名前空間をインポートする同等のオプションがありません。つまり、各.cs ファイル内で「using」キーワードを使用する必要があります。

```
using System;
using Autodesk.Revit.DB;
using Autodesk.Revit.UI;
using Autodesk.Revit.Attributes;
```

1.2 次のコードを1_HelloWorld.csに追加します。これはRevitに最小の外部コマンドを定義するものです:

```
<C#>
/// <summary>
/// Hello World #1 - A minimum Revit external command.
/// </summary>
[Autodesk.Revit.Attributes.Transaction( Autodesk.Revit.Attributes.TransactionMode.ReadOnly)]
public class HelloWorld : Autodesk.Revit.UI.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 Autodesk.Revit.UI.Result.Succeeded;
}
}
</C#>

```

このコマンドクラスは「HelloWorld」と呼ばれ、それは以下のように定義されます:

- コマンドクラスは `IFunction` から派生します。
- `Execute()` メソッドを実装する必要があります。このメソッドは 3 つの引数を持ち、“Succeeded”、“Failed”、“Cancelled” のいずれかの結果を返します。

この例では、Revit スタイルのダイアログである `TaskDialog` を使用して、メッセージを表示します。

最初の行に注目してください: 属性とトランザクションはオプションではありません。これらがないと、Revit はコマンドを認識してくれません。

- トランザクション モード — `Manual`、`ReadOnly` (トランザクションなし) のいずれかでトランザクションの動作をコントロールします。Revit 2016 までは `Automatic` モードがサポートされておりましたが、Revit 2017 で廃止されました。

この時点でコードはビルドする準備が整いました。コードをビルドしてみてください。

2. アドイン・マニフェスト・ファイル

Revitは、コマンドとアプリケーションを登録するためにアドイン・マニフェスト・ファイル(.addin)を使用します。Revitは起動時にマニフェスト・ファイルを自動的に読み取ります。使用するコンピュータにマニフェスト・ファイルを置くことができる2つの場所があります:全ユーザ用とログインユーザ固有の場所です。

Windows 7/8用:

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

使用するシステムとユーザ要件に一致するフォルダを見つけるか、フォルダを作成してください。次に、新しいテキストファイルを作成して、それを「`AdnplIntroCs.addin`」として、下記の内容で .addin ファイルを保存します:

```

<?xml version="1.0" encoding="utf-8"?>
<RevitAddIns>
  <AddIn Type="Command">
    <Text>Hello World</Text>
  </AddIn>
</RevitAddIns>

```

```

<FullClassName>IntroCs.HelloWorld</FullClassName>
<Assembly>C:\...\IntroCs.dll</Assembly>
<AddInId> 8195FDC2-5B44-43D8-9637-51E0967A9562</AddInId>
<VendorId>ADNP</VendorId>
<VendorDescription>Autodesk, Inc. www.autodesk.com</VendorDescription>
</AddIn>
</RevitAddIns>

```

.addin ファイルは、次の情報を含んでいます:

- アドインのタイプ(コマンドかアプリケーション)
- [アドイン]タブ>>[外部ツール]パネルの下に現われるテキスト
- 名前空間を含む完全なクラス名
- dll か アセンブリ・モジュールへのフルパス
- GUID あるいはコマンドの一意的な名前
- ベンダー Id –登録済の[Registered Developer Symbol](#) (RDS)
- ベンダー記述 - 会社名

使用する環境と一致するように適切に調整してください。例えば、上記の例はCSの接尾辞が付きます(名前空間、クラス名、アセンブリのフルパスおよび名前)。 [Revit API 開発者用ガイド](#) の「[アドインの登録](#)」項目には、各タグと多くのタグ・オプションについて記述する一覧が記載されています。新しいGUIDを生成するツールを使用することもできます(Microsoft Visual Studio の [ツール]>[GUIDの生成])。

3. コマンドのテスト

dll のビルドが成功して、適切なパスに .addin が配置されていれば、このコマンドをテストする準備が整ったことになります。

1. Revitを起動して新しいプロジェクトを作成したら、まず最初に[アドイン]タブを見えます(アドインが正しく登録されていないと、アドイン名は表示されません)。
2. [外部ツール]パネルの下に、登録した[Hello World]が表示されるはずです(図4)。

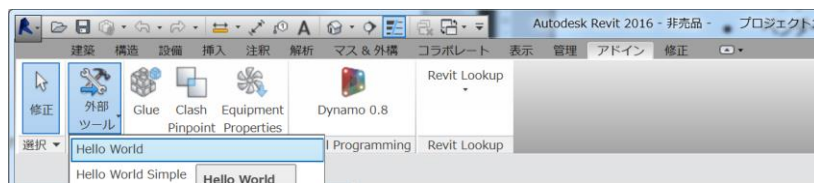


図4.[アドイン]タブの[外部ツール]の下に表示される “Hello World”コマンド

3. コマンドを実行すると、“Hello World!” ダイアログが表示されます(図1)。

4. 名前空間の完全修飾名を省略して「Hello World」

再度、“Hello World” のコードに戻りましょう。ここまでの “Hello World” コードでは、名前空間の完全修飾名を使用していました。下記は名前空間の記述を省略した “Hello World” のコードです。この省略により、タイプ時間を節約してコードをより読み易くすることができます。

```
<C#>
// Hello World #2 - simplified without full namespace.
//
[Transaction(TransactionMode.ReadOnly)]
public class HelloWorldSimple : IExternalCommand
{
    public Result Execute(
        ExternalCommandData commandData,
        ref string message,
        ElementSet elements)
    {
        TaskDialog.Show("My Dialog Title", "Hello World Simple!");

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

このコードを正しく動作させるために、必要な名前空間がインポートされていることを確認してください(VB.NET 内でコードを記述している場合には、プロジェクトプロパティの名前空間のインポートを確認してください)。

1.2 で記述したコードを編集して、再ビルドして確認することができます。

5. 「Hello World App」:外部アプリケーション

最小の外部コマンドを記述する方法と、Revitで外部コマンドを登録する方法を学習しました。ここでは、別のタイプの Revit アドインである 外部アプリケーション の作成方法を解説します。Revit の起動時やシャットダウン時など、アプリケーションレベルで実行する必要がある機能を加えたい場合、外部アプリケーションを使用することができます。例えば、独自のリボン・パネルを[アドイン]タブに加えたければ、Revit 起動時に、リボン・パネルを追加する関数を呼び出します。

5.1 「Hello World App」外部アプリケーション用のコードを追加していきます。
下記は外部アプリケーションの最小のコードです:

```
<C#>
// Hello World #3 - minimum external application
public class HelloWorldApp : IExternalApplication
{
    // OnShutdown() - called when Revit ends.
    public Autodesk.Revit.UI.Result OnShutdown(
```



```

        Autodesk.Revit.UI.UIControlledApplication application)
    {
        return Result.Succeeded;
    }

    // OnStartup() - called when Revit starts.
    public Autodesk.Revit.UI.Result OnStartup(
        Autodesk.Revit.UI.UIControlledApplication application)
    {
        TaskDialog.Show("My Dialog Title", "Hello World from App!");
        return Result.Succeeded;
    }
}
</C#>

```

今回は、**IExternalApplication** からクラスを派生していることに注目してください。オーバーライドすることができる2つの関数があります:

- **OnStartup()** — Revit起動時に呼び出される
- **OnShutdown()** — Revit終了時に呼び出される

この実習では、Revitの起動時に、「Hello World from App!」という文言のメッセージ ダイアログを表示します。

コードは現在のドキュメント(1_HelloWorld.cs)に加えることができます。プロジェクトをビルドして、ビルドエラーが表示されないことを確認してください。

5.2 外部アプリケーションのためのアドイン・マニフェスト・ファイルを追加します。

外部アプリケーション用のアドイン マニフェストは次のとおりです:

```

<AddIn Type="Application">

    <Name>Hello World App</Name>
    <FullClassName>IntroCs.HelloWorldApp</FullClassName>
    <Assembly>C:\...\IntroCs.dll</Assembly>
    <AddInId>49E64932-F4DA-4053-B795-3B4B7AE5C12C</AddInId>
    <VendorId>ADNP</VendorId>
    <VendorDescription>Autodesk, Inc. www.autodesk.com</VendorDescription>
</AddIn>

```

アドイン タイプが “Application” であることを確認して、“<Text>” タグの代わりに “<Name>” タグを使用してください。それ以外のタグは、外部コマンドと同じです。既存の AdnpIntroCs.addin ファイルで以前に記述した <Addin></AddIn> 定義の後に、次のようになるよう記述を加えます。もちろん、お使いの環境に適合させる調節が必要になります。あるいは、それぞれ個別の .addin ファイルに分割することもできます。

```

<?xml version="1.0" encoding="utf-8" standalone="no"?>
<RevitAddIns>

```

```

<AddIn Type="Command">
  <Text>Hello World</Text>
  <FullClassName>IntroCs.HelloWorld</FullClassName>
  <Assembly> C:\...\IntroCs\RevitIntroVB.dll</Assembly>
  <AddInId>0B997216-52F3-412a-8A97-58558DC62D1E</AddInId>
</AddIn>
<AddIn Type="Application">
  <Name>Hello World App</Name>
  <FullClassName>IntroCs.HelloWorldApp</FullClassName>
  <Assembly>C:\...\IntroCs.dll</Assembly>
  <AddInId>49E64932-F4DA-4053-B795-3B4B7AE5C12C </AddInId>
  <VendorId>ADNP</VendorId>
  <VendorDescription>Autodesk, Inc. www.autodesk.com</VendorDescription>
</AddIn>
</RevitAddIns>

```

5.3 アプリケーションをテストする

これでアプリケーションをテストする準備ができました。Revit を起動すると、自動的に「Hello World from App」メッセージが表示されるはず (図5)。

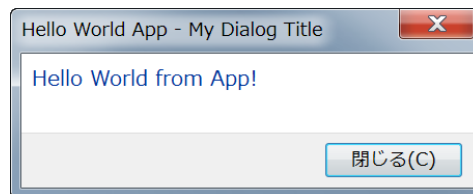


図5.Revit起動時に表示される外部アプリケーションからの対話メッセージ

6. Command Data と Revit オブジェクト・モデル

ここまで Revit の外部コマンドと外部アプリケーションを定義する方法を学習しました。次に、再度外部コマンドに戻って、Execute() メソッドのコマンド引数について見ていきます。Execute() メソッドには、3つの引数があることがわかります:

1. commandData(ExternalCommandData) — Revitモデルにアクセスするエントリー ポイント(入力)
2. message(string) — コマンドが失敗した場合のユーザへのメッセージ (出力)
3. elements(ElementSet) — コマンドが失敗した場合にハイライトしたい要素のリスト(出力)

6.1 デバッガで commandData を検証する

最も重要なオブジェクトは commandDataです。Revit モデルへのアクセスはここからスタートします。このオブジェクトの内容を確認するために、“Hello World” コードの任意の行にブレークポイントを置いて(例えば、TaskDialog.Show()の行)、コマンドを実行します。ブレークポイントにヒットしたら、commandData の内

容を見てください。commandData から参照すれば、どんな種類の情報がそこからアクセス可能か確かめることができます。例えば階層を参照することで、

commandData >> Application >> Application >> VersionXxx

現在実行しているRevitのバージョン名、バージョン番号およびビルド番号を得ることができます。下記の図6および図7でローカルビューのサンプル画像を表示しています。

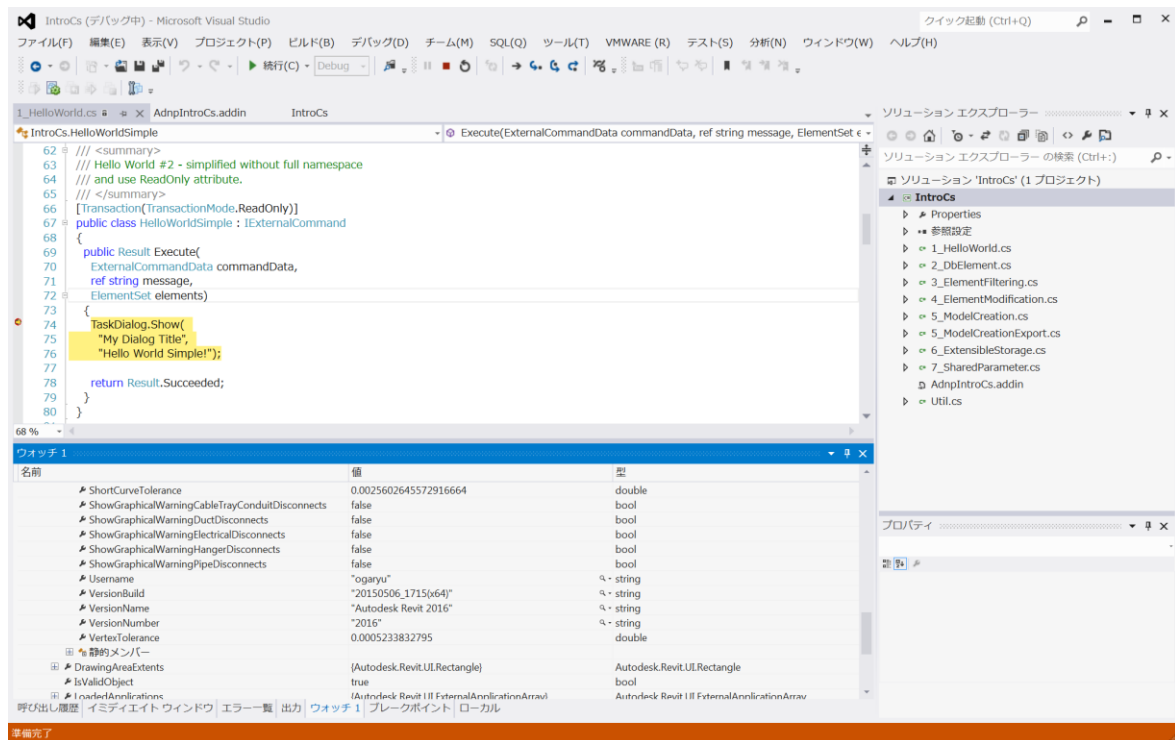


図6.コードの中でcommandDataを検査

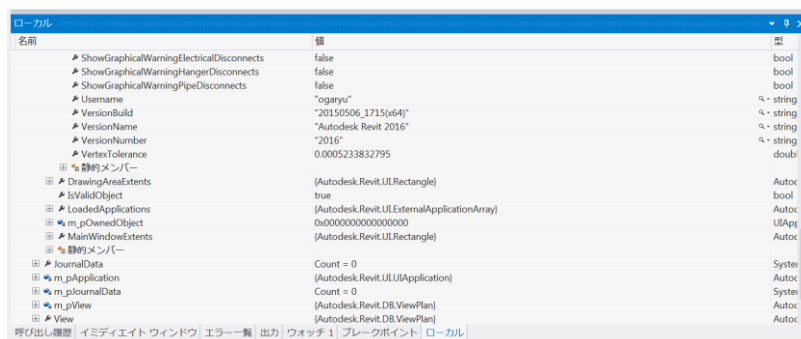


図7.ローカルビュー内でcommandDataを検査

commandData からデータ取得して表示する新しい外部コマンドを定義してみましょう。

6.2 新しい外部コマンドクラスを定義する

例えば、「CommandData」として新しい外部コマンド クラスを指定します。

- コマンドクラス名:**CommandData**

デバッガを使用して、commandData からどんな種類の情報を得ることができるか探ってみてください。次に、取得した情報を表示するコードを記述していきます。

下記のコード例は、バージョン情報および名前と、Revitファイル内の壁タイプを表示するものです:

```
<C#>
// Command Arguments and Revit Object Model
[Transaction(TransactionMode.ReadOnly)]
public class CommandData : IExternalCommand
{
    public Result Execute(
        ExternalCommandData commandData,
        ref string message,
        ElementSet elements)
    {
        // The first argument, commandData, provides access to the top most object model.
        // You will get the necessary information from commandData.
        // To see what's in there, print out a few data accessed from commandData
        //
        // Exercise: Place a break point at commandData and drill down the data.

        UIApplication rvtUiApp = commandData.Application;
        Application rvtApp = rvtUiApp.Application;
        UIDocument rvtUiDoc = rvtUiApp.ActiveUIDocument;
        Document rvtDoc = rvtUiDoc.Document;

        // Print out a few information that you can get from commandData
        string versionName = rvtApp.VersionName;
        string documentTitle = rvtDoc.Title;

        TaskDialog.Show(
            "Revit Intro Lab",
            "Version Name = " + versionName
            + "\nDocument Title = " + documentTitle);

        // Print out a list of wall types available in the current rvt project:

        FilteredElementCollector collector = new FilteredElementCollector(rvtDoc);
        collector.OfClass(typeof(Autodesk.Revit.DB.WallType));

        string s = "";
        foreach (WallType wallType in collector)
        {
            s += wallType.Name + "\r\n";
        }

        // Show the result:
    }
}
```

```

TaskDialog.Show(
    "Revit Intro Lab",
    "Wall Types (in main instruction):\n\n" + s);

// 2nd and 3rd arguments are when the command fails.
// 2nd - set a message to the user.
// 3rd - set elements to highlight.

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

```

コードをビルドして、エラーが表示されないことを確認してください。

6.3 アドイン・マニフェスト・ファイルを追加する

アドイン・マニフェスト・ファイルは既に学習している通り、次のような記述になります (セクション2「アドイン・マニフェスト・ファイル」参照)。

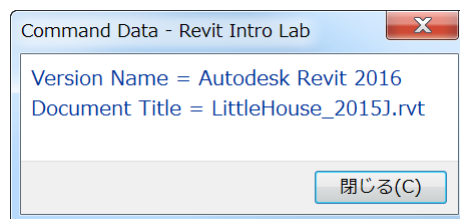
```

<AddIn Type="Command">
  <Text>Command Data</Text>
  <FullClassName> IntroCs.CommandData</FullClassName>
  <Assembly>C:\...\IntroCs.dll</Assembly>
  <AddInId>AA90426A-E8AA-45b4-84AF-861E865871AE</AddInId>
  <VendorId>ADNP</VendorId>
  <VendorDescription>Autodesk, Inc. www.autodesk.com</VendorDescription>
</AddIn>

```

6.4 コマンドの実行とテスト

“Command Data” コマンドを実行すると command Data から取得した情報を表示するダイアログを見ることとができるはずですが。下記の図8は、Revit の「建築テンプレート」(DefaultJPNJPN.rte) テンプレートを使って作成したプロジェクトで、サンプルを実行した際のイメージです。



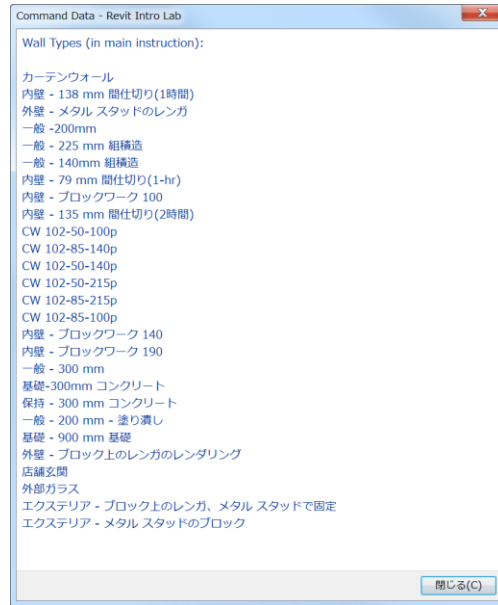


図8.「Command Data」コマンドでExternalCommandDataから情報を表示

7. サマリ

この実習では、次の項目について、Revit APIの基本を学習しました。

- 外部コマンドあるいはコマンド・レベル・アドインを実装する
- 外部アプリケーションあるいはアプリケーションレベル・アドインを実装する
- 外部コマンド、または、外部アプリケーション定義に属性を適用する
- 外部コマンドや外部アプリケーションを Revit に登録するためにアドイン・マニフェスト・ファイルを記述する
- 外部コマンド・データによって Revit モデルにアクセスする

次の実習では、Revit の各要素をより詳細に見ることで、Revit 要素が製品の中でどのように表現されているか学習します。

ツール

次の実習の前に、Revit API の学習やデバッグ時に有用なツールを紹介します。

- **Revit Lookup** — Revit データベース構造を「覗き見る」ことを可能にするツールで、今回の実習を通して利用する Revit API プログラムの必須ツールです。Revit 20xx SDK には Revit Lookup ツールは同梱されていません。下記のページからダウンロードしてご利用ください。

RevitLookup

<https://github.com/jeremytammik/RevitLookup>

- **アドイン・マネージャ** — このツールは、Revit の実行中に dll をロード/ロード解除することを可能するツールで、Revit SDK のサブ フォルダにインストーラとして提供されます。
- **SDKSamples20xx.sln** — /Samples/フォルダで提供されるソリューション ファイルで、より容易にすべてのサンプル・プロジェクトをビルドするために提供される。現在、Revit SDK には 100 を超えるサンプルが提供されているので、すべてのサンプルを個々にビルドすることは退屈なタスクになります。多くのサンプルをビルドして調査する場合には、このソリューション ファイルを開いて Visual Studio で開いてビルドするほうが手軽です。SDKSamples20xx.sln と共に、各サンプル プロジェクトの参照パスを更新する **RevitAPIDllsPathUpdater.exe** も提供されています。Revit が、お使いのコンピュータでデフォルト位置とは異なったフォルダにインストールされている場合や、業種別の Revit 製品を使用している場合に便利です。
- **RvtSamples** — /Samples/フォルダで提供されるサンプルのうちの 1 つです。このサンプルは、[アドイン]リボンタブの下に [RvtSamples] パネルを作成して、ExternalApplications を除く他のすべてのサンプルを統合表示します。さまざまな SDK サンプルのテストに便利です。

外部アプリケーションである Revit Lookup ツールを Revit に追加して使用できるようにしてみましょう。

Autodesk Developer Network