SD11192 - Advanced Revit® Code Refactoring - C# Language Features and Design Patterns Can Help

Dave Echols
Senior Programmer
Class summary

This class will explain how code based on the Revit® API can be refactored to simplify usage, isolate functionality, and improve maintainability. A detailed look will be taken at extension methods, the Action and Func delegate classes, Microsoft LINQ, and design patterns.
At the end of this class, you will be able to:

- Learn how to add extension methods to existing Revit® API objects to add functionality
- Learn how to utilize Action class delegates combined with lambda expressions to encapsulate functionality and simplify code
- Learn how to create LINQ expressions to select and process operations on groups of objects
- Learn how to use design patterns to separate code responsibilities and provide uniform code interfaces
Introductions
Inspiration

SD5215: Sharpen Your C# Code for AutoCAD

Key Learning

- Learn how to use delegates to reduce duplicate code
- Understand how LINQ can be used with the AutoCAD software API
- Learn how to apply abstraction and dependency injection into your C# code
- Learn how to use Windows Presentation Foundation (WPF) and the Model-View-ViewModel design pattern
Instruction Manuals Outdated?

Visit: AutodeskUniversity.com

Click on My AU to start building your own desk reference (with materials from this decade).
Who Am I?

Let me introduce myself:

- Senior Programmer at Hankins & Anderson, Inc.
- Hankins & Anderson is an Autodesk Developer Network member.
- I have used and/or developed on various CADD systems since 1986.
- I have worked with AutoCAD® since 1988.
- I have worked with Revit® since 2009.
- I have brewed beer at home since 1982.
Who Are You?

Let's get to know each other:

- How many of you are an Autodesk Developer Network member?
- How many of you have used any of the functionality we will cover in class with your Revit® Add-Ins?
- How many of you have developed for Revit® for 3 years?
- How many of you have developed for Revit® for 5 years?
- How many of you have developed for Revit® longer than 5 years?
- Are there any fellow home brewers here today?
What It Is

What you can expect:

- Clear, researched definitions for the main topics of this class
- My interpretation of how these features can be used with the Revit® API
- Concrete examples from Hankins & Andersons’s code base
- A starting point for you to create and implement your own ideas
- Code examples included in class files
What It Is Not

What you will not get:

- All of the source code from Hankins & Andersons’s code base
- All of the other possibilities of using these features with the Revit® API
- Detailed advice on how these features might work in your Revit® solutions
- Programming nirvana
Extension Methods
Definition

From the Microsoft Developer Network C# Programming Guide

“Extension methods enable you to "add" methods to existing types without creating a new derived type, recompiling, or otherwise modifying the original type.” *

public static void DoWorkOnDocument(this Document revitDoc)
{
    revitDoc.GetElement(uniqueId);
}

Classes Extended

Classes from the Revit® API that have been extended:

- BasicFileInfo
- Document
- ViewSheet
- ModelPath
- Transaction
- TransactionGroup
- FilteredElementCollector
- …
public static bool IsOpenedAsCentral(this BasicFileInfo basicInfo, 
                                     string documentPathName) 
{ 
    string fileName = Path.GetFileName(documentPathName).ToUpper(); 
    string centralFile = string.Empty; 
    if (basicInfo.CentralPath.Length > 0) 
    { 
        centralFile = Path.GetFileName(basicInfo.CentralPath).ToUpper(); 
    } 
    return centralFile.Equals(fileName); 
}
public static string GetVersionNumber(BasicFileInfo basicInfo) {
    if (basicInfo == null) return string.Empty;
    if (basicInfo.SavedInVersion.IndexOf("2013") > -1) return "2013";
    // 2014 & 2015
    if (basicInfo.SavedInVersion.IndexOf("2016") > -1) return "2016";
    Debug.Assert(false, "Error");
    return string.Empty;
}
public static bool IsValidForPrintingOrExport(ViewSheet viewSheet)
{
    if (viewSheet.IsPlaceholder) return false;
    if (viewSheet.GetAllPlacedViews().Count > 0) return true;
    if (viewSheet.HasSchedules()) return true;
    if (viewSheet.HasTextNotes()) return true;
    if (viewSheet.HasAnnotationSymbols()) return true;
    if (!viewSheet.CanBePrinted) return false;
    return false;
}
public static TransactionStatus EndTransaction(this Transaction transaction,
    TransactionType transactionType,
    FailureHandlingOptions options = null)
{
    TransactionStatus status = transaction.GetStatus();
    if (status != TransactionStatus.Started) return status;
    switch (transactionType) {
        case TransactionType.Commit:
            return options != null ? transaction.Commit(options)
                : transaction.Commit();
        case TransactionType.RollBack:
            return options != null ? transaction.RollBack(options)
                : transaction.RollBack();
    }
    return TransactionStatus.Error;
Action & Func Delegates with Lambda Expressions
Definitions (Action<[T], …> Delegate)

From the Microsoft Developer Network .NET Framework Class Library

“Encapsulates a method that has no parameters and does not return a value.” *


From the MSDN Blog of Bruno Terkaly, Principal Software Engineer, Microsoft

“You can use the Action(Of T) delegate to pass a method as a parameter without explicitly declaring a custom delegate. The sugar here is you don’t have to declare a delegate. The compiler is smart enough to figure out the proper types.” **

Definitions (Func<[T], …, TResult> Delegate)

From the Microsoft Developer Network .NET Framework Class Library

“Encapsulates a method that has no parameters and returns a value of the type specified by the TResult parameter.” *


From the MSDN Blog of Bruno Terkaly, Principal Software Engineer, Microsoft

“You can use this delegate to represent a method that can be passed as a parameter without explicitly declaring a custom delegate. The encapsulated method must correspond to the method signature that is defined by this delegate.” **

Definitions (Lambda Expressions)

From the Microsoft Developer Network C# Programming Guide

“A lambda expression is an anonymous function that you can use to create delegates or expression tree types. By using lambda expressions, you can write local functions that can be passed as arguments or returned as the value of function calls. Lambda expressions are particularly helpful for writing LINQ query expressions.”

Transaction transaction = null;
try {
    transaction = new Transaction(revitDoc, "SingleImport");
    if (transaction.Start("SingleImport") !=
        TransactionStatus.Started) return;
    DoWorkHere();
    DoMoreWorkHere();
    DoEvenMoreWorkHere();
    transaction.Commit();
} catch (Exception ex) { transaction.RollBack(); }
finally { if(transaction != null) transaction.Dispose(); }
using (Transaction transaction = new Transaction(revitDoc, "SingleImport"))
{
    try
    {
        if (transaction.Start("SingleImport") != TransactionStatus.Started) return;
        DoWorkHere();
        DoMoreWorkHere();
        DoEvenMoreWorkHere();
        transaction.Commit();
    }
    catch (Exception ex) { transaction.Rollback(); }
}
using (Transaction transaction = new Transaction(revitDoc, "SingleImport"))
{
    try
    {
        if (transaction.Start("SingleImport") != TransactionStatus.Started) return;
        DoAllTheWorkHere();
        transaction.Commit();
    }
    catch (Exception ex) { transaction.RollBack(); }
}
public static void UsingTransaction(Document revitDoc) 
{ 
  using (var transaction = new Transaction(revitDoc, "SingleImport")) 
  { 
    try 
    { 
      if(transaction.Start("SingleImport") != TransactionStatus.Started) return;
        DoAllTheWorkHere();
        transaction.Commit();
    } 
    catch (Exception ex) 
    { 
      transaction.RollBack(); 
      throw; 
    } 
  } 
}
Uses With The Revit® API - Transactions

```csharp
public static void UsingTransaction(this Document revitDoc, Action<Transaction> action)
{
    using (var transaction = new Transaction(revitDoc, "SingleImport"))
    {
        try
        {
            if (transaction.Start("SingleImport") != TransactionStatus.Started) return;
            action(transaction);
            transaction.Commit();
        }
        catch (Exception ex)
        {
            transaction.Rollback();
            throw;
        }
    }
}
```
```csharp
public static TransactionStatus UsingTransaction(this Document revitDoc,
                                             Action<Transaction, TransactionType, FailureHandlingOptions> action,
                                             TransactionType transactionType = TransactionType.Commit,
                                             FailureHandlingOptions failureHandlingOptions = null,
                                             string transactionName = null)
{
    transactionName = SetTransactionName(transactionName);
    using (var transaction = new Transaction(revitDoc, transactionName))
    {
        try
        {
            TransactionStatus transactionStatus = transaction.Start(transactionName);
            if (transactionStatus != TransactionStatus.Started) return transactionStatus;
            action(transaction, transactionType, failureHandlingOptions);
            return transaction.EndTransaction(transactionType, failureHandlingOptions);
        }
        catch (Exception ex)
        {
            transaction.EndTransaction(TransactionType.RollBack, failureHandlingOptions);
            throw;
        }
    }
}
```
private void GeneratePdfFiles(string folder, List<ElementId> ids)
{
    try
    {
        string fileName = Path.GetFileNameWithoutExtension(RevitDoc.PathName);
        Manager = new PlotManager(RevitDoc);
        Setup = new PlotSetup(Manager);
        RevitDoc.UsingTransaction(TransactionType.RollBack, "Print PDF Command", (transaction, transactionType) => GeneratePdfFiles(folder, ids, fileName));
    }
    catch(Exception ex)
    {
        // Log errors and inform user
    }
}
LINQ
Definition

From LINQ: .NET Language-Integrated Query

“.NET Language-Integrated Query defines a set of general purpose standard query operators that allow traversal, filter, and projection operations to be expressed in a direct yet declarative way in any .NET-based programming language.”

LINQ Resources

LINQ Resources for basic reference and Revit® API use:

- http://www.tutorialsteacher.com/linq/linq-tutorials
- http://thebuildingcoder.typepad.com/blog/filters/
- http://www.linqpad.net/
LINQ Expression Usage

LINQ Expressions used in Hankins & Anderson’s code:

- Ribbon – RevitUI
- Filters
Working with the Revit® User Interface

This code uses the assembly ADWINDOWS.dll
LINQ Query Syntax

```csharp
public RibbonTab GetRibbonTab(string tabId){
    var selectedTab = from tab in ComponentManager.Ribbon.Tabs
                      where tab.Id.ToUpper().Equals(tabId.ToUpper())
                      select tab; return selectedTab.First();}

public RibbonPanel GetTabPanel(RibbonTab tab, string automationName){
    var selectedPanel = from panel in tab.Panels
                         where panel.Source.AutomationName.ToUpper().
                              Equals(automationName.ToUpper())
                         select panel; return selectedPanel.First();}

public RibbonItem GetPanelItem(RibbonPanel panel, string id){
    var selectedItem = from item in panel.Source.Items
                       where item.Id.ToUpper().Equals(id.ToUpper())
                       select item; return selectedItem.First();}
```
LINQ Method Syntax

```csharp
public RibbonTab GetRibbonTabX(string tabId){
    return ComponentManager.Ribbon.Tabs.First(tab =>
    tab.Id.ToUpper().Equals(tabId.ToUpper()));
}

public RibbonPanel GetTabPanelX(RibbonTab tab, string name){
    return tab.Panels.First(panel =>
    panel.Source.AutomationName.ToUpper().
    Equals(name.ToUpper()));
}

public RibbonItem GetPanelItemX(RibbonPanel panel, string id){
    return panel.Source.Items.First(item =>
    item.Id.ToUpper().Equals(id.ToUpper()));
}
```
public static class ApplicationInitializedHandler
{
    public static void Handler(object sender, ApplicationInitializedEventArgs e)
    {
        RibbonTab tab = RevitUi.Instance.GetRibbonTab("Insert");
        RibbonPanel linkPanel = RevitUi.Instance.GetTabPanel(tab, "Link");

        RibbonItem revitLinkItem = RevitUi.Instance.GetPanelItem(linkPanel,
            "ID_RVTDOC_LINK");
        revitLinkItem.IsEnabled = false;

        RibbonItem ifcLinkItem = RevitUi.Instance.GetPanelItem(linkPanel,
            "ID_IFC_LINK");
        ifcLinkItem.IsEnabled = false;
    }
}
using (FilteredElementCollector collector =
    new FilteredElementCollector(revitDoc))
{
    using (ElementClassFilter viewSheetFilter =
        new ElementClassFilter(typeof(ViewSheet)))
    {
        var selectedViewSheets = from vs in collector
            .WherePasses(viewSheetFilter)
            .Cast<ViewSheet>()
            where vs.SheetNumber.StartsWith("E2")
            select vs;

        foreach (ViewSheet sheet in selectedViewSheets)
        {
            Debug.Print(sheet.SheetNumber);
        }
    }
}
using (FilteredElementCollector viewCollector =
    new FilteredElementCollector(revitDoc)) {
    foreach (ElementId viewId in viewCollectorOfClass(typeof(ViewDrafting)).
        ToElementIds()) {
        using (FilteredElementCollector collector =
            new FilteredElementCollector(revitDoc)) {
            var selectedTextNotes = from tn in collectorOfClass(typeof(TextNote)).
                OwnedByView(viewId).Cast<TextNote>().
                Where(tn.TextNoteType.Name.IndexOf("1/8") > -1)
                .Select(tn);
        foreach (TextNote textNote in selectedTextNotes) {
            // Change to 3/32"}
        }
    }
}
Design Patterns
From “Design Patterns: Elements of Reusable Object-Oriented Software”
“A design pattern names, abstracts and identifies the key aspects of a common design structure that makes it useful for creating a reusable object-oriented design.”

Design Patterns used in Hankins & Anderson’s code:

- Singleton – RevitUI, TaskDialogManager, GlobalSettings, CommandFactory
- Command – PrintPdfCommand, DwgToDraftingViewCommand
- Factory – CommandFactory, ViewFactory
- Adapter – HaDetailArc, HaDetailEllipse, HaDetailLine
- Builder - PanelBuilder
Uses With The Revit® API – Command & Factory

- IExternalCommand
- Command
- Execute

- <<Creator>>
- CommandFactory
- Register
- Execute
- Unregister

- «Implementation»
- PrintPdfCommand
- Execute

- <<Interface>>
- ICommand
- Execute

- <<Base>>
- SynchronousCommandBase
- Execute

- Uses With The Revit® API – Command & Factory
- #AU2015
public Result Execute(ExternalCommandData data, ref string msg, ElementSet elements) 
{try 
    { 
      using (PrintPdfForm frm = new PrintPdfForm(Assembly.GetExecutingAssembly().GetName(), revitDoc)) 
      { 
        CommandFactory.GetInstance().Register(typeof (PrintPdfCommand), "Print PDF Files"); 
        CommandFactory.GetInstance().Execute("Print PDF Files", 
          new object[] {commandData, elements, 
            frm.OutputFolder, 
            frm.SelectedSheets}); 
        CommandFactory.GetInstance().Unregister(typeof (PrintPdfCommand), "Print PDF Files"); 
      } 
      return Result.Succeeded; 
    } 
    catch (Exception ex) { message = ex.Message; } return Result.Failed;
public sealed class CommandFactory
{
    private static readonly CommandFactory _instance = new CommandFactory();

    private CommandFactory()
    {
        Commands = new List<ICommand>();
        RegisteredCommands = new List<ICommand>();
    }

    public static CommandFactory GetInstance()
    {
        return _instance;
    }
    // ...
}
Uses With The Revit® API - Adapter

«Implementation»
HaDetailLine
- StartPoint
- EndPoint
- LineStyleName
- LineStyleUniqueId
- IsInitialized
- UniqueElementId
- Create()
- Initialize()
- GetDetailLine()

VS

«Implementation»
DetailLine
- Parameters

«Implementation»
Line
- CreateBound()

«Implementation»
Document
- Create
public static HaDetailLine Create(XYZ startPoint, XYZ endPoint, View currentView){
    HaDetailLine newHaDetailLine = new HaDetailLine();
    using (Line newLine = Line.CreateBound(startPoint, endPoint)){
        using (DetailLine detailLine =
            currentView.Document.Create.NewDetailCurve(currentView, newLine) as DetailLine){
            if (detailLine != null){
                newHaDetailLine._detailLine = detailLine;
                newHaDetailLine.IsInitialized = true;
            }
        }
    }
    return newHaDetailLine;
}
private bool DrawLine(AcadEntity entity, AcadEntity parentEntity, XYZ startPoint, XYZ endPoint){
    try {
        HaDetailLine newLine = HaDetailLine.Create(startPoint, endpoint, ActiveView);
        newLine.AssignLineStyle(GlobalSettings.ImportLineWeights[GetColorIndex(entity, parentEntity)], HaLineStyles.GetLinePatternType(entity, parentEntity));
        return true;
    } catch (ElementCreationException ex) { // Log Errors }
    catch (Exception ex) { // Log Errors }
    return false;}

#AU2015
Be heard! Provide AU session feedback.

- Via the Survey Stations, email or mobile device.
- AU 2016 passes awarded daily!
- Give your feedback after each session.
- Give instructors feedback in real-time.
Questions & Answers