Facing the Elephant in the Room: Making Autodesk® Revit® Add-ins That Cooperate with Worksharing

Scott Conover – Autodesk

DV1888

The Autodesk Revit API offers the ability to build a wide variety of customizations that drive the contents of projects, elements, and views. Typically the add-in covers the singular case where a single user is operating against the project. However, many projects are large and Revit software work is workshared, with multiple users editing and modifying the project simultaneously. It is important for Revit add-in developers to design their add-ins to "play nicely" in workshared environments. This can include ensuring elements are checked out before trying to modify them, updating and synchronizing changes with central, and opening documents correctly depending on the purpose of the add-in. This class focuses on techniques that can make a well-behaved and useful add-in in the workshared environment, with particular focus on new APIs that have been introduced in Revit 2014. It also covers the similarities and differences in techniques when dealing with server-based worksharing. Attendees should have knowledge of C# and Revit API. Prior experience with Revit worksharing is also helpful.

Learning Objectives

At the end of this class, you will be able to:

- Explain the basics of Revit worksharing techniques and terminology, especially features that affect add-in development
- Design a Revit add-in to work well in a workshared environment
- Determine the right way to open, read, and/or modify a workshared document for your purposes
- Describe how to deal with both file-based worksharing and server-based worksharing from your add-in

About the Speaker

Scott is a Software Development Manager in the Autodesk® Revit® development team, focused on API and Interoperability. Since joining Autodesk in 2007, he has led a team working on the design, implementation and testing of the rapid expansion of the Autodesk® Revit® API.

Scott has 15 years of experience producing Application Programming Interfaces for parametric 3D modeling systems in a variety of languages and styles. His primary focus has been to enable customers to automate repetitive tasks, extend the application user interface, and transfer data between the application and different data formats.

Scott holds a Master of Computer Systems Engineering degree from Northeastern University with a concentration on CAD/CAM/CAE.

Scott.Conover@autodesk.com

About this course

This course is focused on 2014 enhancements to the Revit API related to View and Schedule capabilities. For completeness, it also covers some, but not all, 2013 and earlier capabilities. It is not intended to be comprehensive review of earlier APIs. For that, see AU course CP3133 presented by Steven Mycynek last year, or the Revit API documentation and samples.

The focus of this course is the sample code uploaded as additional class materials. Techniques in the samples are presented as "best recommendations" not "Autodesk requirements". In some cases, these are not proven to work in shipping add-ins (to my knowledge). These may not be the only way to construct a successful add-in.

Worksharing Basics examples

WorksharingAPIBasicsCommands.cs

GetWorksetInfoCommand	Shows information about the active workset and the workset of the active view.
NotifyWorksetOnNewElement	The action to take when documents change indicating the newly added elements and what workset they are added to.
MakeChanges	Demonstrates creation of an element that ends up in the document's active workset.
MakeChangesAndChangeElementWorkset	Demonstrates creation of an element that is changed to belong to a different workset.
GetElementAndElementIdInfo	Demonstrates how caches in memory of ElementId can be affected by Worksharing operations.
ToggleWorksetGraphics	Demonstrates how to activate different worksharing display modes.

Worksharing editing examples

WorksharingAPIEditingCommands.cs

TryToEditProjectInfo	Tries to edit the ProjectInfo for the document, and optionally uses a failure preprocessor to rollback if there is a Worksharing error.
TryToEditGeometricElement	Tries to edit a geometric element, and optionally uses a failure preprocessor to rollback if there is a Worksharing error.
AttemptToCheckoutInAdvance	Tries to check out the given element before editing.

Extensible storage editing examples

ExtensibleStorageCommands.cs

AddAddInInfoElement	Adds or updates the element that stores Add-in wide info to the document. This technique is one step better than adding all the add-in wide data to a global document element such as ProjectInfo.
ShowAddInInfoElement	Shows the contents of the add-in wide info element, if it exists.
AddWallAddInInfoElement, AddColumnAddInInfoElement, AddRoomAddInInfoElement	Adds or updates the contents of one of the type-specific add-in info results elements. This technique is better than storing all data in one application-wide element because different local users working on different aspects of the add-in can update their data without editing conflicts.
ShowWallAddInInfoElement, ShowColumnAddInInfoElement,	Shows contents of the type-specific add-in info results elements.

ShowRoomAddInInfoElement	

Updater examples

WorksharingAPIUpdaterCommands.cs

TriggerWallBulkUpdateCommand	Triggers an updater to make changes to walls as system changes. This allows the changes to be made even if the walls are checked out for editing elsewhere, and does not require the walls to be checked out locally. Note that the changes to the walls will be lost if the walls were edited by another user before these changes are synchronized with central. The updater is set as optional to avoid marking the model as requiring it. The trigger for the updater is creation of a Data Storage element with a particular name - that element will be deleted by the updater.
WallBulkUpdater	The Updater class that processes the wall changes.
EnableCalculationUpdaterCommand	Enables the updater that depends on a calculation related to model elements. Also adds the corresponding DocumentChangedEvent subscription used to detect and trigger updater changes from worksharing events such as synch with central or reload latest.
WorksharedUpdater	An Updater that executes a calculation related to model elements. The updater counts the number of walls in the model and includes this result in the text of a note.

Worksharing operations examples

WorksharingAPIOperationsCommands.cs

SynchWithCentralWithMessage	Synchronize with central, but only after showing a message to the user to confirm that it is OK to do so. It includes an option for relinquishing all or keeping worksets checked out.
ReloadLatestWithMessage	Reloads the latest with central, but only after confirming with the user that it is OK to do so.
OpenDetached	Opens a non-graphical document detached, and then opens a graphically activated documented detached with a prompt about whether or not to preserve worksets.
OpenLastViewed	Opens the document while opening only the last viewed worksets.
OpenNewLocalFromDisk	Creates a new local from a central file on disk, and opens it.
OpenNewLocalFromServer	Creates a new local from a central file in Revit server, and opens it.
CopyAndOpenDetached	Copies a Revit server model locally and opens it detached. This is the workflow for applications that need to execute read-only operations on the server model where no local is required.
CreateLinkToServerModel	Creates a Revit link to a model located on the Revit server.
FindWSAPIModelPathOnServer	Uses the Revit Server REST API to recursively search the folders of the Revit Server for a particular model.