AUTODESK UNIVERSITY

SD20868

Revit Usage and Model-Data Reporting Simplified with C#

Don Rudder
STG Design

Learning Objectives

- Gain insight into your model and user patterns using C#
- Learn how to report user actions in real-time without causing latency
- Learn how to report targeted model conditions in real time
- Learn how to extract detailed model data quickly and efficiently

Description

See how Revit software can provide powerful insight to your model contents, and how your design team is building them through external data reporting. Information is most useful when presented in a format that is easy to explore and analyze. This class will demonstrate several techniques for getting information out of your models and into a format that you can more easily digest and visualize, all with the goal of helping you make better decisions. This session features Revit, Revit Architecture, and Revit MEP. AIA Approved

Don Rudder

Don Rudder is a Building Information Modeling (BIM) manager with an MEP (mechanical, electrical, and plumbing) design background and a strong software-development skill set. With over 10 years in mechanical and electrical design, he has served as BIM/CAD manager for an additional 8 years. Rudder's software-development skills are primarily centered around, but not limited to, the architecture, engineering, and construction industry, and he's a published author.

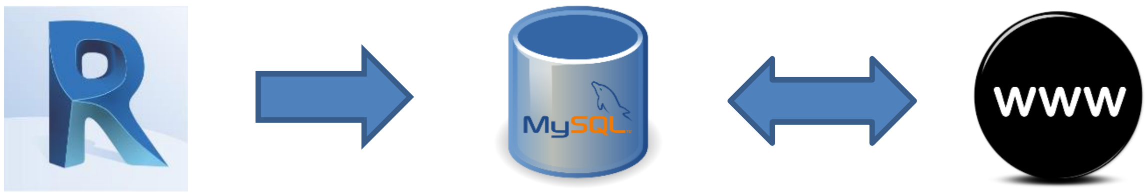
Introduction

The audience that this document was targeted for includes designers with Revit experience that are not necessarily professional fulltime software developers. Not all designers are software developers – so this class will hopefully help bridge that gap for designers that want to take on a software development skillset.

A working knowledge of Revit Add-Ins is assumed.

Gain insight into your model and user patterns using C#

The big picture problem that we are trying to solve in this class is provide a simple means to visualize reliable information for Revit users and models within an organization. An ASP.NET web site will be the interface used to visualize the information that we harvest from Revit using an events based Revit Add-In. A MySQL database will be used to store all of the information that will drive the system.



A Few Tips to Avoid Latency

There are a few things that you should avoid at all costs so that your solution doesn't add any unwanted delays for your users. Since the Revit API limits you to a single thread, you won't be able to use threads to analyze the model and do something with the resulting data without disturbing the user. Be courteous with your data mining techniques and avoid long running tasks. Break your data mining tasks up into small pieces and perform.

Avoid Making REST Calls to External Services

You may from time to time need to make a REST call to an external service, but limiting these calls for POST and PUT will help out your users tremendously. Using an external executable that you can call asynchronously will help keep your solution from getting a bad rap with your users.

Stay Focused

Stay focused on your goals and be careful with what your event processes. If you have long running code that is not worth the loss in productivity and time for your organization, remove it and keep your code running as efficiently as possible.

Test, Test, Test

Always test your code thoroughly prior to distribution. This should be obvious, but is an easy step to skip for an overly confident developer.

Writing the Local Data Files

The best way to avoid user permission errors for writing files is to use a folder under the user's profile. I typically like to store user specific data in the following folder:

%USERPROFILE%\AppData\Local\<My Application Name>\...

This is typically the most efficient way to write data out and keep the latency under control for an application. You could opt to post the files to a network location, but this could get messy if you have laptop users or users that may be in varying distances from the central network location. Posting to an FTP also opens up a door for problems for the same reasons.

A code snippet used to get and create if necessary the target directory where we will store the small json files on the user's machine.

```
/// <summary>
/// Path to store data: %USERPROFILE%\AppData\Local\SD20868\ [Events] , [Sessions]>
/// </summary>
/// <param name="isEvent">True for regular events, false for Revit session</param>
/// <returns></returns>
internal static string DataPath(bool isEvent)
{
  try
  {
    string m_folder = isEvent ? "Events" : "Sessions";
    var m_path = Path.Combine(
      Environment.GetFolderPath(
        Environment.SpecialFolder.LocalApplicationData),
      "SD20868",
      m_folder);
    if (!Directory.Exists(m_path)) Directory.CreateDirectory(m_path);
    return m_path;
  } catch (Exception ex)
  {
    return "";
  }
}
```

We can then use a method to write the data for an object in json format as shown below.

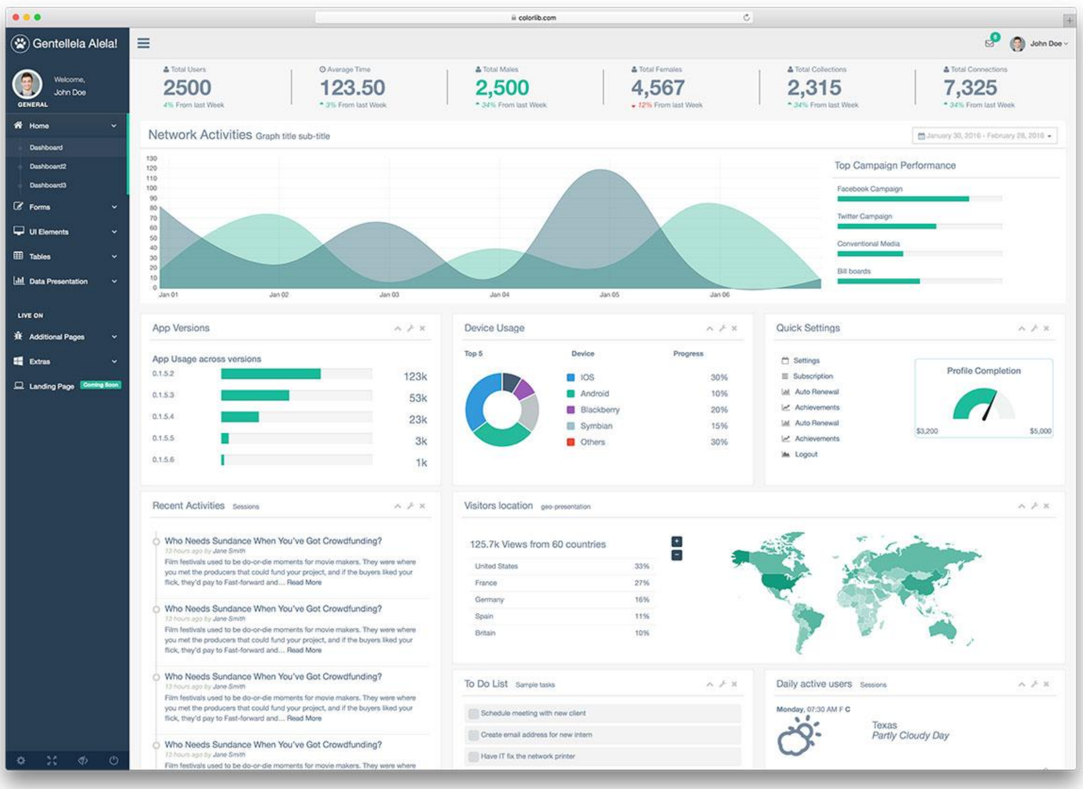
```
/// <summary>
/// Write an Object to Json
/// </summary>
/// <param name="sourceData">Top level object to write out to file</param>
/// <param name="fileKind">command kind to use in file name</param>
/// <returns>TRUE if file exists after main function</returns>
/// <remarks></remarks>
internal static bool SaveJson(object sourceData, string fileKind)
{
  try
  {
    // Write Folder Exists?
    var m_eventPath = DataPath(true);
    if (!Directory.Exists(m_eventPath))
    {
      ShowTaskDialog(
        "Missing Path",
        "Failed to Create or Find Directory",
        m_eventPath);
      return false;
    }

    string m_fileName = String.Format("{0} {1} {2}.json",
      Environment.UserName,
      fileKind,
      DateTime.Now.ToString("yyMMdd hhmmss"));
    string m_finalPath = Path.Combine(m_eventPath, m_fileName);
    using (StreamWriter sw = new StreamWriter(m_finalPath, false))
    {
      sw.WriteLine(JsonConvert.SerializeObject(sourceData));
    }
    return File.Exists(m_finalPath);
  }
  catch {}
  return false;
}
```

The Web Dashboard

The web dashboard is an ASP.NET web site with the backend code built in C#. We will use javascript to make calls to a REST API for our charting objects.

The Open Source Gentella Web Template

I opted to use an open source and free to use web template as a starting point for the main dashboard. Colorlib is the author of this MIT Licensed template and is available on Github: https://github.com/puikinsh/gentelella



0.1.5.2	123k
0.1.5.3	53k
0.1.5.4	23k
0.1.5.5	3k
0.1.5.6	1k

Top 5

Device	Progress
IOS	30%
Android	10%
Blackberry	20%
Symbian	15%
Others	30%

Who Needs Sundance When You've Got Crowdfunding?
13 hours ago by Jane Smith
Film festivals used to be do-or-die moments for movie makers. They were where you met the producers that could fund your project, and if the buyers liked your flick, they'd pay to Fast-forward and... Read More

Who Needs Sundance When You've Got Crowdfunding?
13 hours ago by Jane Smith
Film festivals used to be do-or-die moments for movie makers. They were where you met the producers that could fund your project, and if the buyers liked your flick, they'd pay to Fast-forward and... Read More

Who Needs Sundance When You've Got Crowdfunding?
13 hours ago by Jane Smith
Film festivals used to be do-or-die moments for movie makers. They were where you met the producers that could fund your project, and if the buyers liked your flick, they'd pay to Fast-forward and... Read More

Who Needs Sundance When You've Got Crowdfunding?
13 hours ago by Jane Smith
Film festivals used to be do-or-die moments for movie makers. They were where

125.7k Views from 60 countries

United States	33%
France	27%
Germany	16%
Spain	11%
Britain	10%

- Schedule meeting with new client
- Create email address for new intern
- Have IT fix the network printer

GENTELLA TEMPLATE

This template will already have most everything that we'll need in terms of widgets and charts. It also helps that it already looks good and has a clean and organized layout structure that uses the Twitter Bootstrap standards that are so prevalent throughout web development these days. We will need to perform a few minor tweaks on the template so that it works with ASP.NET technology.

Central MySQL Database

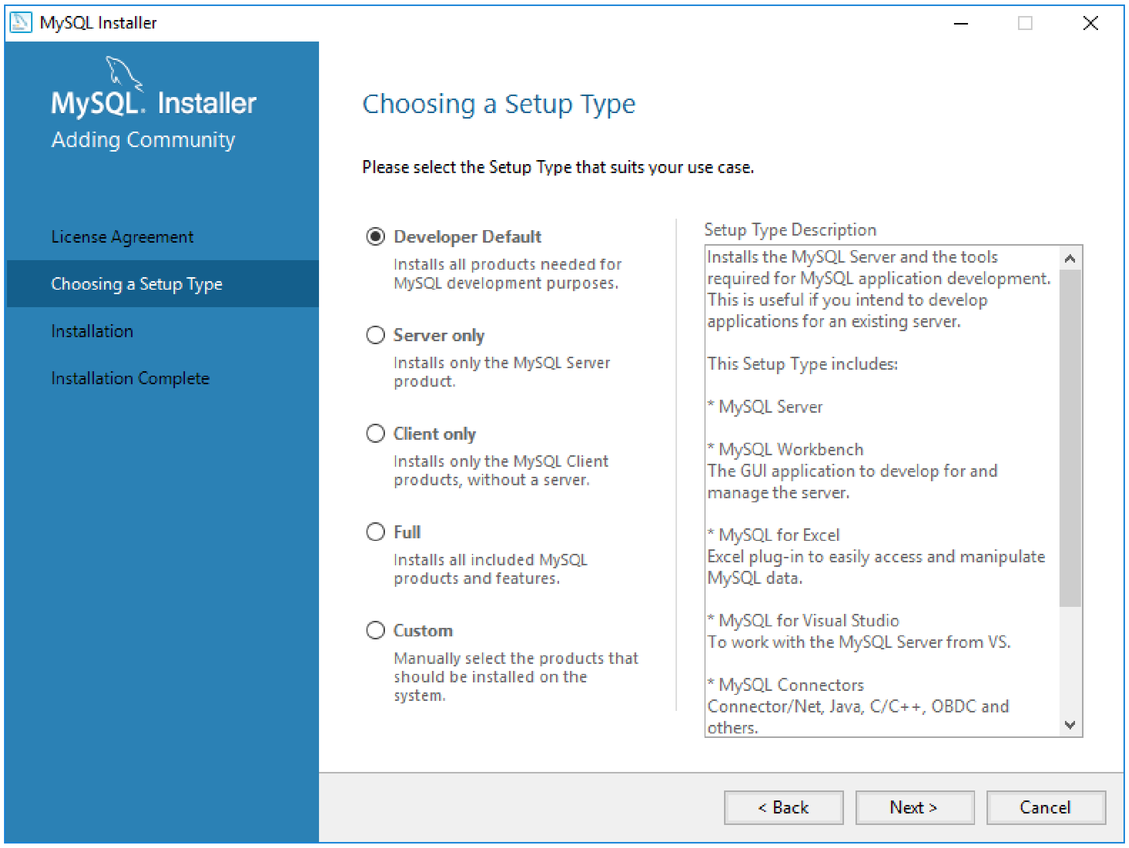
A MySQL database will be at the center of our solution for storing all of the necessary information that we harvest from Revit. MySQL Community Server is free and ideal for this kind of application.

Download and Install MySQL

First download MySQL 5.7 from this link:
http://dev.mysql.com/get/Downloads/MySQLInstaller/mysql-installer-web-community-5.7.16.0.msi

You will need administrative privileges to install MySQL. Initiate the installation by double clicking the msi file that you downloaded in the previous step. This is a web install and will be downloading a large amount of data, so plan for this part to take longer if you have a slower internet connection.

Choose “Developer Default” when prompted for a Setup Type. This will make sure that you get everything necessary to run the samples provided in this class.



Choosing a Setup Type

Please select the Setup Type that suits your use case.

[✓ Developer Default] Installs all products needed for MySQL development purposes.

Server only — Installs only the MySQL Server product.

Client only — Installs only the MySQL Client products, without a server.

Full — Installs all included MySQL products and features.

Custom — Manually select the products that should be installed on the system.

Setup Type Description

Installs the MySQL Server and the tools required for MySQL application development. This is useful if you intend to develop applications for an existing server.

This Setup Type includes:

* MySQL Server

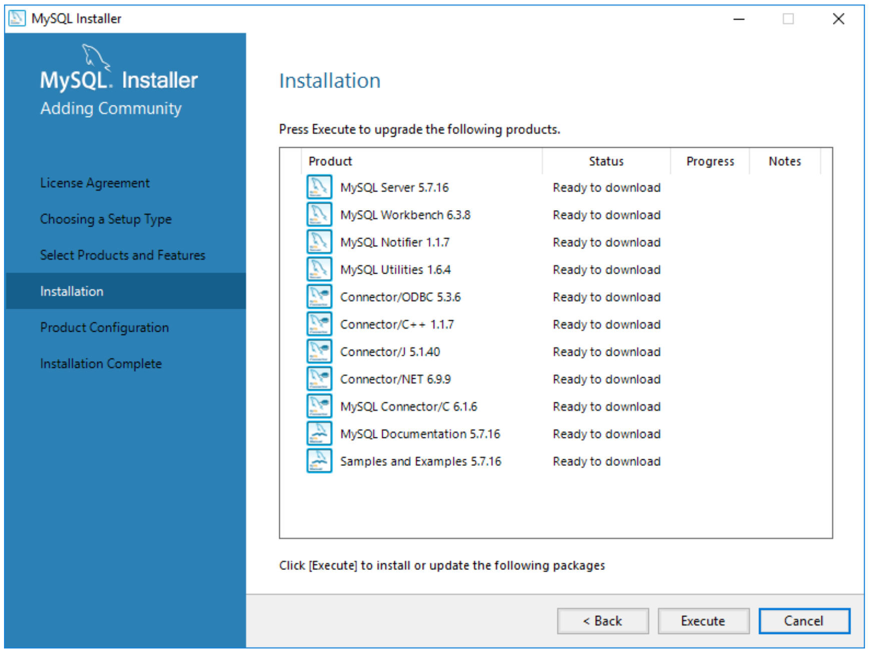
* MySQL Workbench
The GUI application to develop for and manage the server.

* MySQL for Excel
Excel plug-in to easily access and manipulate MySQL data.

* MySQL for Visual Studio
To work with the MySQL Server from VS.

* MySQL Connectors
Connector/Net, Java, C/C++, OBDC and others.

The list of components that will install are shown in the Installation dialog. Click the "Execute" button to start the download and installation of each component.



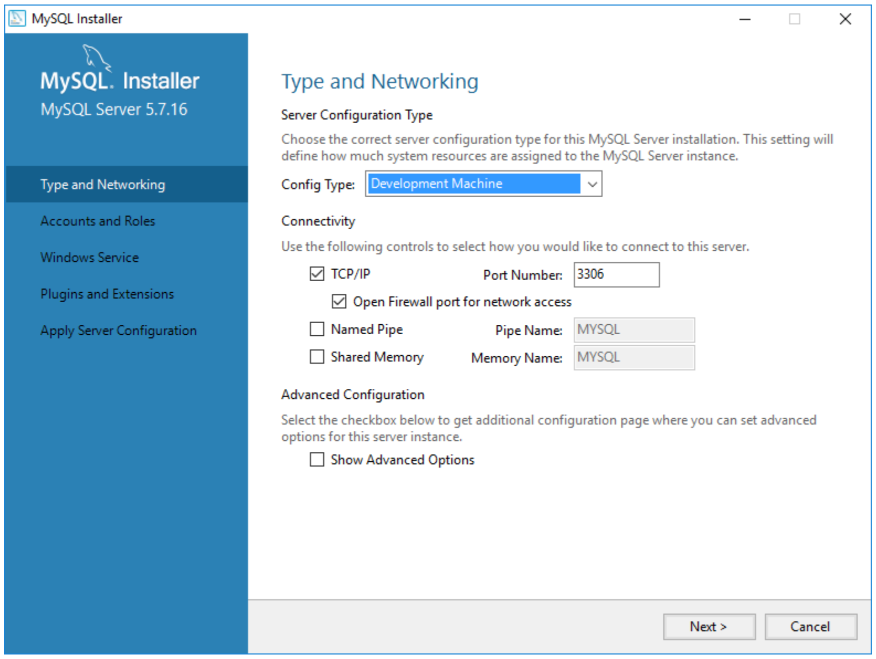
Installation

Press Execute to upgrade the following products.

Product	Status	Progress	Notes
MySQL Server 5.7.16	Ready to download		
MySQL Workbench 6.3.8	Ready to download		
MySQL Notifier 1.1.7	Ready to download		
MySQL Utilities 1.6.4	Ready to download		
Connector/ODBC 5.3.6	Ready to download		
Connector/C++ 1.1.7	Ready to download		
Connector/J 5.1.40	Ready to download		
Connector/NET 6.9.9	Ready to download		
MySQL Connector/C 6.1.6	Ready to download		
MySQL Documentation 5.7.16	Ready to download		
Samples and Examples 5.7.16	Ready to download		

Click [Execute] to install or update the following packages

Accept the defaults when prompted for Type of Networking.



Type and Networking

Server Configuration Type

Choose the correct server configuration type for this MySQL Server installation. This setting will define how much system resources are assigned to the MySQL Server instance.

Config Type: Development Machine

Connectivity

Use the following controls to select how you would like to connect to this server.

[✓] TCP/IP   Port Number: 3306

[✓] Open Firewall port for network access

[ ] Named Pipe   Pipe Name: MYSQL

[ ] Shared Memory   Memory Name: MYSQL

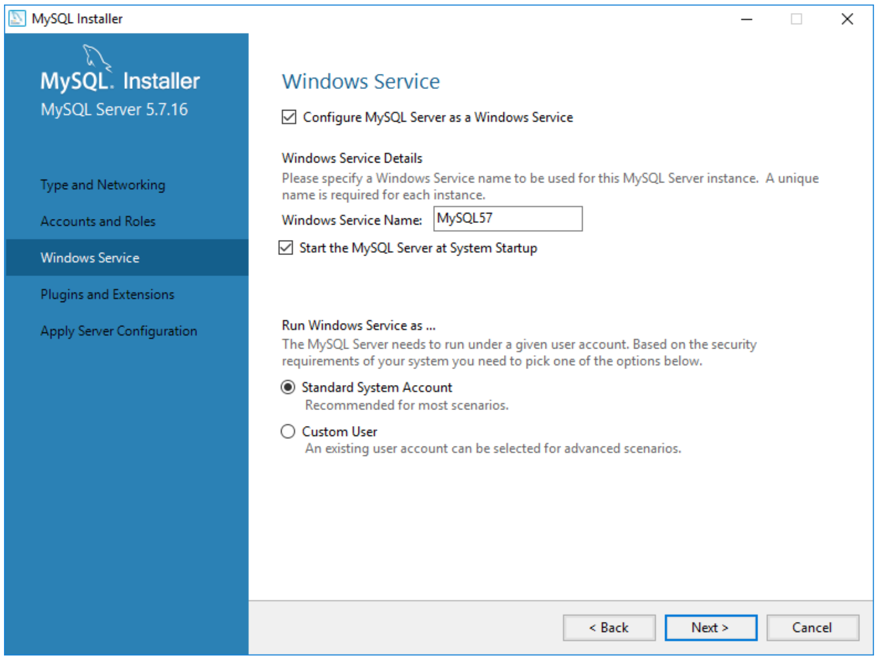
Advanced Configuration

Select the checkbox below to get additional configuration page where you can set advanced options for this server instance.

[ ] Show Advanced Options

The materials provided in this dataset will assume that your user "root" has a password of "root" without quotes. If your root password changes, you'll need to hunt down and update this password in the provided source code.

Accept the defaults when prompted with the Windows Service dialog.



Windows Service

☑ Configure MySQL Server as a Windows Service

Windows Service Details
Please specify a Windows Service name to be used for this MySQL Server instance. A unique name is required for each instance.

Windows Service Name: MySQL57

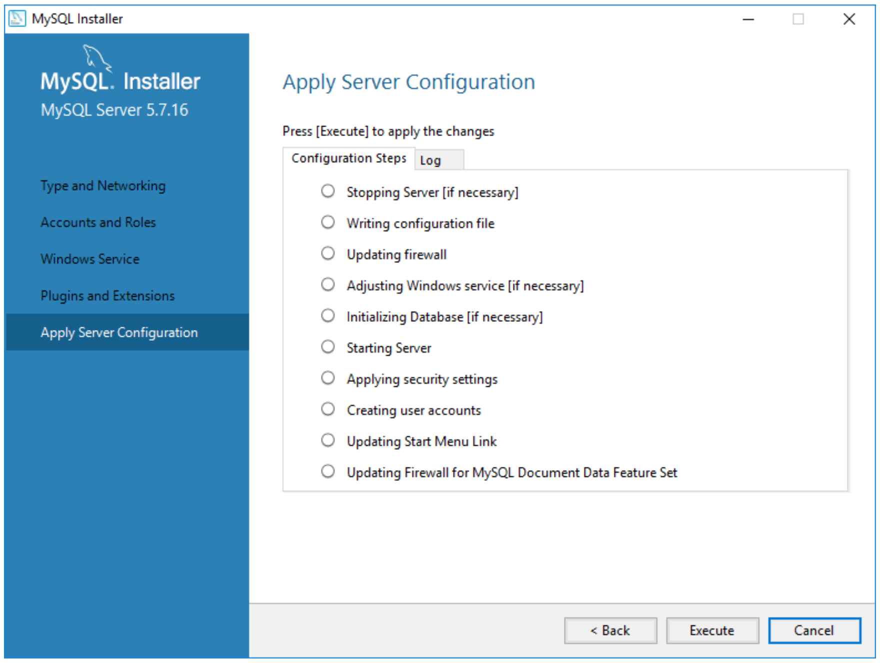
☑ Start the MySQL Server at System Startup

Run Windows Service as ...
The MySQL Server needs to run under a given user account. Based on the security requirements of your system you need to pick one of the options below.

● Standard System Account
Recommended for most scenarios.

○ Custom User
An existing user account can be selected for advanced scenarios.

Next click “Execute” to apply the configurations to your MySQL installation.



Apply Server Configuration

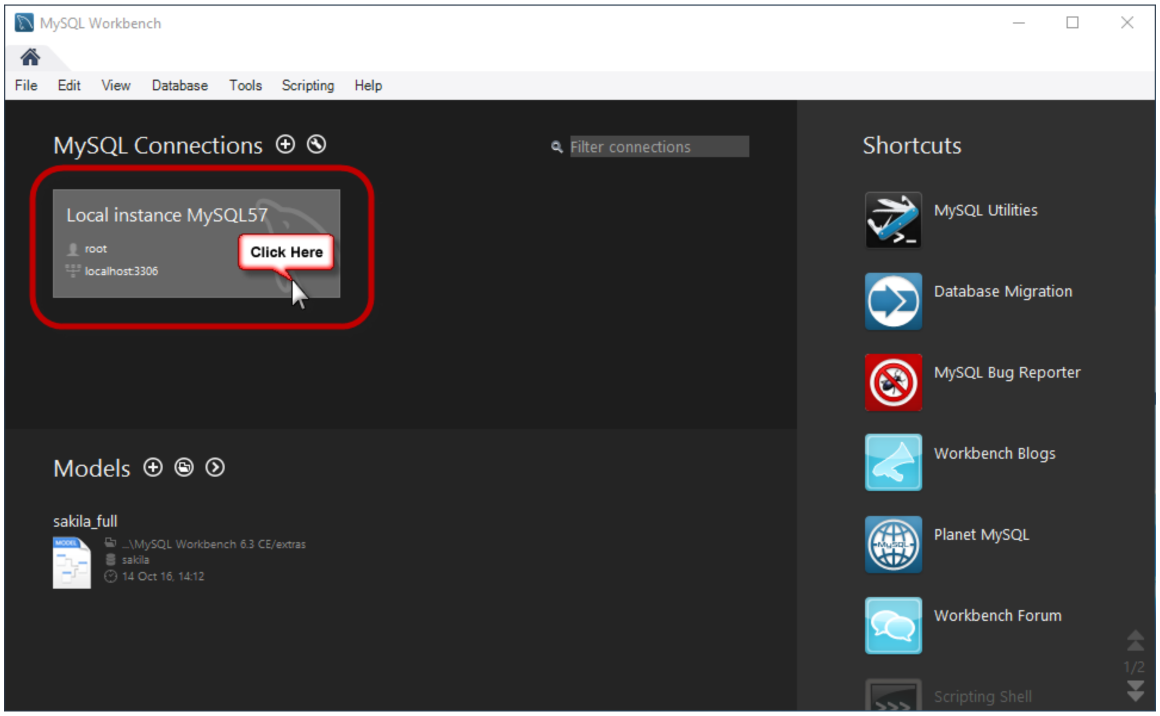
Press [Execute] to apply the changes

○ Stopping Server [if necessary]
○ Writing configuration file
○ Updating firewall
○ Adjusting Windows service [if necessary]
○ Initializing Database [if necessary]
○ Starting Server
○ Applying security settings
○ Creating user accounts
○ Updating Start Menu Link
○ Updating Firewall for MySQL Document Data Feature Set

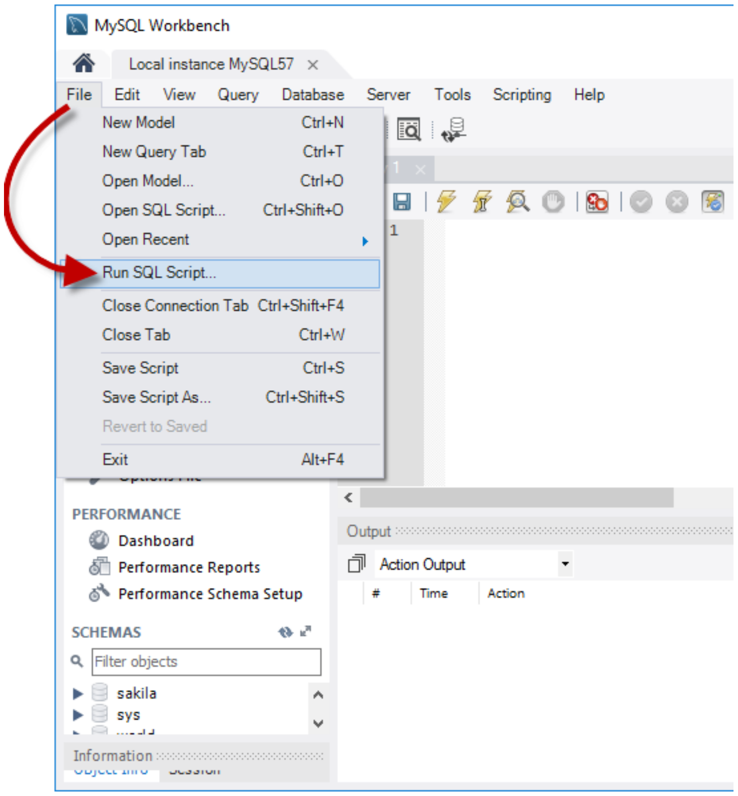
Install the Sample Database

There is a file included in the sample dataset named “SD20868_MySQL.sql.” This is an empty version of the database that you can use to start with a fresh and clean dashboard system.

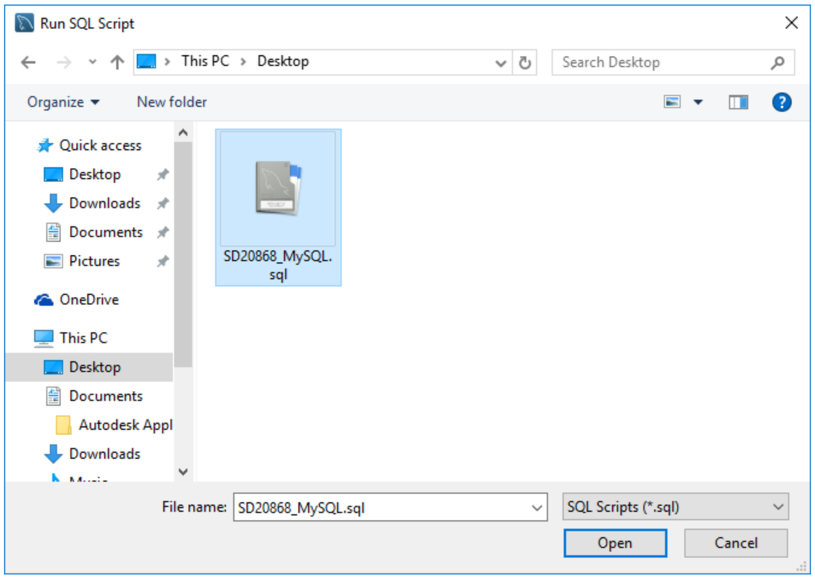
Launch the MySQL Workbench utility and open your local instance by clicking on the "Local instance MySQL57" tile as indicated below.



Click on File and then choose "Run SQL Script"



Next navigate to the folder where you extracted the files contained in the dataset sample and chose the file named “SD20868_MySQL.sql” and click “Open.”



File name: SD20868_MySQL.sql

You should now have the sample database loaded onto your system.

The Revit Add-In

A Revit Add-In will be used to pull the data that we’re interested in out from Revit and an externally defined database posting executable will watch over a specified folder for new json files as they get written to the user’s local disk and post that data into the central database.

The Add-In in does nothing more than subscribe to various events that we’re interested in and perform a few light operations on the model when those events are triggered. The results get written out to json files.

The External Data Posting Application

Since making REST calls can be time consuming, we delegate these tasks to an external application to handle this for us. Our sample will utilize a direct database post method, but could easily be modified to target a REST API system and provide the same results.

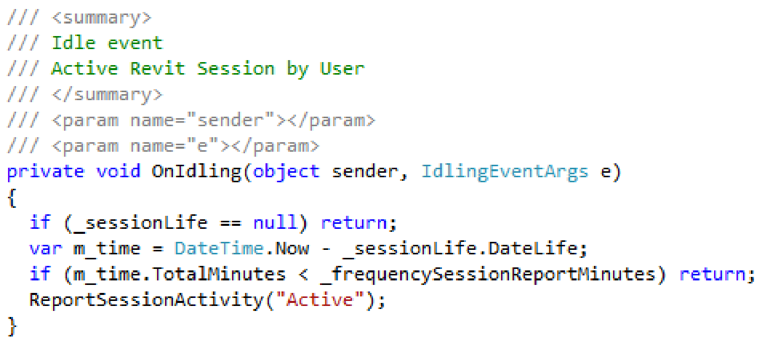
Learn how to report user actions in real-time without causing latency

It is important that you take extreme caution when implementing a data tracking system inside Revit. Avoiding time consuming tasks so that your designers don't have to wait on your utility is paramount. Delays for your users basically cost the company money and you need to take every precaution necessary to avoid that.

This section is dedicated to providing insight as to what your users are doing in the models. What models they are in, what views they are using, and what command they are executing. We also want to understand how many elements they are adding, editing, and deleting. We'll also watch over the Levels category and be able to know who deleted a level if it happens.

Revit Session Monitoring

Being able to reliably see how many sessions are active in your organization can be quite useful. We can use the OnIdle event to report the activity of a specific session to accomplish this. We can then quickly understand who has how many sessions open and for how long.



```
/// <summary>
/// Idle event
/// Active Revit Session by User
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void OnIdling(object sender, IdlingEventArgs e)
{
  if (_sessionLife == null) return;
  var m_time = DateTime.Now - _sessionLife.DateLife;
  if (m_time.TotalMinutes < _frequencySessionReportMinutes) return;
  ReportSessionActivity("Active");
}
```

Learn how to report targeted model conditions in real time

We'll walk through how to subscribe to a few Revit API events and get some useful data out that we can send over to a database and in turn be able to visualize this data in our web dashboard.

The data that we will target for real-time analysis will include:

- Command Names Executed
- Document Create, Open, Close, Save, Save-as
- Sync with Central
- View Activations
- When Revit Sessions Start, Close, and remain Active
- Groups
- In-Place Families

Target Data

We will focus on a few specific data points and do everything we can to avoid causing any latency for the users as they trigger the events that will run our data mining tasks.

Targeting Specific Deleted Elements and Command Names Executed

Being able to monitor and understand exactly when specific model elements get deleted can help prevent ongoing damage to a design model. The category of elements that are suitable for this kind of monitoring include Levels, Ceilings, and Links. Our sample dashboard will show hot to monitor the Levels category and show us who deleted a level and when it was deleted.

The event that we will subscribe to watch over the deletion of elements is the ***DocumentChanged*** event. We will subscribe to this event from within our implementation of ***IExternalApplication***. The line of code below subscribes to this event by adding a listener method named ***OnDocumentChanged*** where we will implement all of our code for dealing with this event.

```
uiConApp.ControlledApplication.DocumentChanged += OnDocumentChanged;
```

This event will also be used to log the commands that users are accessing.

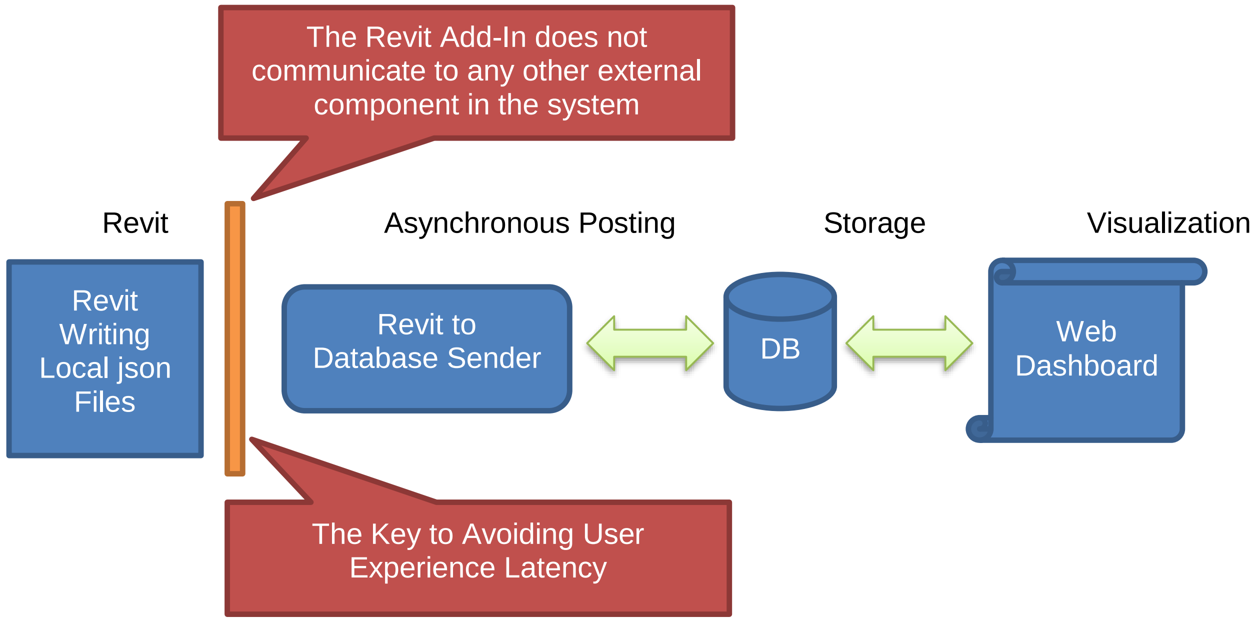
Learn how to extract detailed model data quickly and efficiently

The key with getting the data out of Revit quickly and efficiently is to break up what you are targeting into a series of smaller tasks. Don't try to get everything you are after all at once when you are using other events in your Revit Add-In to watch over other parts of the model.

If you are wanting to watch over the Levels category, use the document open and Sync with Central event handlers to update that information. Don't use the document idle for tasks that can take a long time to complete running or don't necessarily need to be revisited as frequently because they don't tend to change that often at all.

Sending Data Without Interfering with your Users

The key with sending data to your central database without bothering your users is to first store the data into a series of small, quick to write json files and then implement an external executable that you can call asynchronous from Revit to post the data. This will prevent latency for your users in Revit. If the external application takes a little longer to post the final data to the central system, it won't be noticed and this is ideal.


The Revit Add-In does not communicate to any other external component in the system
Revit
Asynchronous Posting
Storage
Visualization
Revit Writing Local json Files
Revit to Database Sender
DB
Web Dashboard
The Key to Avoiding User Experience Latency