Integration: **Integration MongoDb to Revit**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Reflection;
using System.Windows.Media.Imaging;
using Autodesk.Revit.UI;
using System.IO;

namespace exportcloudMDB
{
   /// <summary>
   /// Implements the Revit add-in interface IExternalApplication
   /// </summary>
   [Autodesk.Revit.Attributes.Transaction(Autodesk.Revit.Attributes.TransactionMode.Manual)]

[Autodesk.Revit.Attributes.Regeneration(Autodesk.Revit.Attributes.RegenerationOption.Manual)]
   public class Application : IExternalApplication
   {
      /// <summary>
      /// Implements the OnShutdown event
      /// </summary>
      /// <param name="application"></param>
      /// <returns></returns>
      public Result OnShutdown(UIControlledApplication application)
      {
         return Result.Succeeded;
      }

      /// <summary>
      /// Implements the OnStartup event
      ///
      /// </summary>
      /// <param name="application"></param>
      /// <returns></returns>
      public Result OnStartup(UIControlledApplication application)
      {
         RibbonPanel panel = RibbonPanel(application);
         AddSplitButtonGroup(panel);


         return Result.Succeeded;
      }
```

```
    /// <summary>
    /// Adds split button group with two push buttons.
    /// One to export door data and the other to import door data
    /// </summary>
    /// <param name="panel"></param>
    private void AddSplitButtonGroup(RibbonPanel panel)
    {
      string thisAssemblyPath = Assembly.GetExecutingAssembly().Location;

      SplitButtonData sbData = new SplitButtonData("SplitGroup", "SplitGroup");
      SplitButton splitButton = panel.AddItem(sbData) as SplitButton;

      PushButtonData exportData = new PushButtonData("ExportData", "ExportData",
thisAssemblyPath, "cloudDB.ExportCommand");
      PushButton exportButton = splitButton.AddPushButton(exportData);
      exportButton.ToolTip = "Export Revit Door Data";

      Uri exportUri = new Uri(Path.Combine(Path.GetDirectoryName(thisAssemblyPath),
"Resources", "mongoExp.ico"));
      BitmapImage expBitmapImage = new BitmapImage(exportUri);
      exportButton.LargeImage = expBitmapImage;

      splitButton.AddSeparator();

      PushButtonData importData = new PushButtonData("ImportData", "ImportData",
thisAssemblyPath, "exportcloudMDB.ImportCommand");
      PushButton importButton = splitButton.AddPushButton(importData);
      importButton.ToolTip = "Import Revit Door Data";

      Uri importUri = new Uri(Path.Combine(Path.GetDirectoryName(thisAssemblyPath),
"Resources", "mongoImp.ico"));
      BitmapImage impBitmapImage = new BitmapImage(importUri);
      importButton.LargeImage = impBitmapImage;

    }

    /// <summary>
    /// Function creates Eduardo Ibg tab and cloudDB ribbon panel
    /// </summary>
    /// <param name="a"></param>
    /// <returns name="ribbonPanel"> </returns>
    public RibbonPanel RibbonPanel(UIControlledApplication a)
    {
      // Tab name
      string tab = "Eduardo Ibg";
      // Empty ribbon panel
      RibbonPanel ribbonPanel = null;
```

```
        //Create ribbon Tab
        try
        {
           a.CreateRibbonTab(tab);
        }
        catch (Exception ex)
        {
           TaskDialog.Show(ex.Message, ex.ToString());
        }


        //Create ribbon panel
        try
        {
           RibbonPanel panel = a.CreateRibbonPanel(tab, "cloudDB");
        }
        catch (Exception ex)
        {
           TaskDialog.Show(ex.Message, ex.ToString());
        }

        // Search existing tab for your panel.
        List<RibbonPanel> panels = a.GetRibbonPanels(tab);
        foreach (RibbonPanel p in panels.Where(p => p.Name == "cloudDB"))
        {
           ribbonPanel = p;
        }

        return ribbonPanel;
     }


  }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using RestSharp;
using System.Net;

namespace exportcloudMDB
{
  public class DoorAPI
```

```
  {
    /// <summary>
    /// HTTP access constant to toggle
    /// between local and cloud server.
    /// </summary>
    public static bool useCloudServer = false;

    /// <summary>
    /// Base url for local testing and cloud url for production
    /// </summary>
    const string baseUrlLocal = "http://localhost:8080/";
    const string baseUrlCloud = "https://mongorevit.herokuapp.com/";

    /// <summary>
    /// Base url for local testing and cloud url for production
    /// </summary>
    public static string RestAPIBaseUrl
    {
      get { return useCloudServer ? baseUrlCloud : baseUrlLocal; }
    }

    /// <summary>
    /// GET JSON data from
    /// the specified mongoDB collection.
    /// </summary>
    public static List<Door> Get(string collectionName)
    {
      RestClient client = new RestClient(RestAPIBaseUrl);
      RestRequest request = new RestRequest("/api" + "/" + collectionName, Method.GET);

      IRestResponse<List<Door>> response = client.Execute<List<Door>>(request);

      return response.Data;
    }

    /// <summary>
    /// Batch POST JSON document data into
    /// the specified mongoDB collection.
    /// </summary>
    public static HttpStatusCode PostBatch(out string content, out string errorMessage,
      string collectionName, List<Door> doorData)
    {
      RestClient client = new RestClient(RestAPIBaseUrl);
      RestRequest request = new RestRequest("/api" + "/" + collectionName + "/" + "batch",
Method.POST);
      request.RequestFormat = DataFormat.Json;
      request.AddBody(doorData);
```

```
            IRestResponse response = client.Execute(request);
            content = response.Content;
            errorMessage = response.ErrorMessage;

            return response.StatusCode;
        }

    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Autodesk.Revit.DB;

namespace exportcloudMDB
{
    public class DoorData : Door
    {
        /// <summary>
        /// DoorData constructor to generate the data to
        /// serialise for REST POST request.
        /// </summary>
        public DoorData(Element door)
        {
            //Get Door Finish parameter
            Parameter doorFinish = door.get_Parameter(BuiltInParameter.Door_Finish);
            string dFinish;

            //Check if Door Finish Paramater has a value
            if (doorFinish.HasValue == true)
            {
                dFinish = doorFinish.AsString();
            }
            else
            {
                dFinish = "";
            }

            _id = door.UniqueId;
            FamilyType = door.Name;
            Mark = door.get_Parameter(BuiltInParameter.All_Model_Mark).AsString();
            DoorFinish = dFinish;
        }
```

```
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace exportcloudMDB
{
    /// <summary>
    /// Door class with get/set autoproperties
    /// </summary>
    public class Door
    {
        public string _id { get; set; }
        public string FamilyType { get; set; }
        public string Mark { get; set; }
        public string DoorFinish { get; set; }

    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Autodesk.Revit.DB;
using Autodesk.Revit.UI;
using Autodesk.Revit.Attributes;

namespace exportcloudMDB
{
    /// <summary>
    /// Implements the Revit add-in interface IExternalCommand
    /// </summary>
    [Transaction(TransactionMode.Manual)]
    public class ImportCommand : IExternalCommand
    {
        public Result Execute(ExternalCommandData commandData, ref string message, ElementSet
elements)
        {
            //Get the Current Session / Project from Revit
            UIApplication uiApp = commandData.Application;

            //Get the Current Document from the Current Session
            Document doc = uiApp.ActiveUIDocument.Document;
```

```
        //REST request to GET door data
        List<Door> doors = DoorAPI.Get("doors");

        //Set Door finish values using unique
        if(doors != null && doors.Count > 0)
        {
           using(Transaction trans = new Transaction(doc, "Import Door Data"))
           {
              trans.Start();

              foreach(Door door in doors)
              {
                 string uId = door._id;
                 Element element = doc.GetElement(uId);

                 string dFinish = door.DoorFinish;
                 Parameter doorFinish = element.get_Parameter(BuiltInParameter.DOOR_FINISH);

                 doorFinish.Set(dFinish);
              }

              trans.Commit();
           }
        }

        TaskDialog.Show("Import Door Data", "Door Data successfuly imported");

        return Result.Succeeded;
     }

  }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Autodesk.Revit.DB;

namespace exportcloudMDB
{
   public class DoorData : Door
   {
      /// <summary>
      /// DoorData constructor to generate the data to
      /// serialise for REST POST request.
      /// </summary>
      public DoorData(Element door)
      {
         //Get Door Finish parameter
         Parameter doorFinish = door.get_Parameter(BuiltInParameter.Door_Finish);
         string dFinish;

         //Check if Door Finish Paramater has a value
         if (doorFinish.HasValue == true)
         {
            dFinish = doorFinish.AsString();
         }
         else
         {
            dFinish = "";
         }

         _id = door.UniqueId;
         FamilyType = door.Name;
         Mark = door.get_Parameter(BuiltInParameter.All_Model_Mark).AsString();
         DoorFinish = dFinish;
      }

   }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Autodesk.Revit.DB;

namespace cloudDB
```

```
{
   public class DoorData : Door
   {
      /// <summary>
      /// DoorData constructor to generate the data to
      /// serialise for REST POST request.
      /// </summary>
      public DoorData(Element door)
      {
         //Get Door Finish parameter
         Parameter doorFinish = door.get_Parameter(BuiltInParameter.DOOR_FINISH);
         string dFinish;

         //Check if Door Finish Paramater has a value
         if (doorFinish.HasValue == true)
         {
            dFinish = doorFinish.AsString();
         }
         else
         {
            dFinish = "";
         }

         _id = door.UniqueId;
         FamilyType = door.Name;
         Mark = door.get_Parameter(BuiltInParameter.All_Model_Mark).AsString();
         DoorFinish = dFinish;
      }

   }
}
```