Revit API Intro Labs

Lab 7 – Shared Parameter

Created by A. Nagy, May 2011
Updated by DevTech AEC WG
Last modified: 11/4/2023

<C#>C# Version**</C#>**

Objective: In this lab, we will learn how to create shared parameters. We'll learn how to:

- Create shared parameter group
- Create shared parameter definition
- Bind it to a specific category – e.g. doors

The following is the breakdown of step by step instructions in this lab:

1. Define a New External Command
2. Create some helper functions
3. Create shared parameter for doors
4. Create per document parameter
5. Summary

1. Define A New External Command

We'll add another external command to the current project.

1.1 Add a new file and define another external command to your project. Let's name them as follows:

- File name: **7_SharedParameter.vb (or .cs)**
- Command class name: **SharedParameter**

Required Namespaces:
In addition to the name spaces you have used, add the name space:

- System.IO

Also declare some constants that hold the properties of the parameter we are creating.

<C#>
```
    const string kSharedParamsGroupAPI = "API Parameters";
    const string kSharedParamsDefFireRating = "API FireRating";
    const string kSharedParamsPath = "C:\\temp\\SharedParams.txt";
```

```
    public Result Execute(
      ExternalCommandData commandData,
      ref string message,
      ElementSet elements)
    {
      UIDocument uidoc = commandData.Application.ActiveUIDocument;
      Application app = commandData.Application.Application;
      Document doc = uidoc.Document;
</C#>
```

2. Create some helper functions

First we need a function that gets back or if not already available then creates a new shared parameters file

```
<C#>
    public static DefinitionFile GetSharedParamsFile(Application app)
    {
      // Get current shared params file name
      string sharedParamsFileName;
      try
      {
        sharedParamsFileName = app.SharedParametersFilename;
      }
      catch (Exception ex)
      {
        TaskDialog.Show(
            "Get shared params file", "No shared params file set:" +
            ex.Message);
        return null;
      }

      if (0 == sharedParamsFileName.Length
         || !System.IO.File.Exists(sharedParamsFileName))
      {
        StreamWriter stream;
        stream = new StreamWriter(kSharedParamsPath);
        stream.Close();
        app.SharedParametersFilename = kSharedParamsPath;
        sharedParamsFileName = app.SharedParametersFilename;
      }

      // Get the current file object and return it
      DefinitionFile sharedParametersFile;
      try
      {
        sharedParametersFile = app.OpenSharedParameterFile();
      }
      catch (Exception ex)
      {
        TaskDialog.Show(
            "Get shared params file", "Cannnot open shared params file:"
            + ex.Message);
```

```
        sharedParametersFile = null;
      }
      return sharedParametersFile;
    }
</C#>
```

We need a function that creates if a specific parameter groups already exists, and if not, then creates it and passes it back to us.

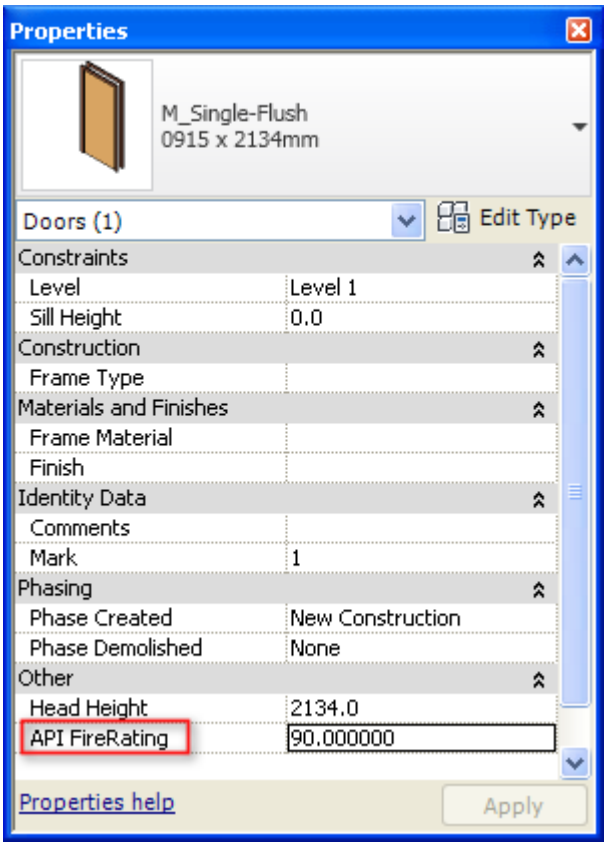
```
<C#>
    public static DefinitionGroup GetOrCreateSharedParamsGroup(
     DefinitionFile sharedParametersFile,
     string groupName)
    {
      DefinitionGroup g = sharedParametersFile.Groups.get_Item(groupName);
      if (null == g)
      {
        try
        {
          g = sharedParametersFile.Groups.Create(groupName);
        }
        catch (Exception)
        {
          g = null;
        }
      }
      return g;
    }
</C#>
```

Then we need a function that checks if a given parameter definition already exists, and if not, then creates it and passes it back to us

```
<C#>
    public static Definition GetOrCreateSharedParamsDefinition(
      DefinitionGroup defGroup,
      ParameterType defType,
      string defName,
      bool visible)
    {
      Definition definition = defGroup.Definitions.get_Item(defName);
      if (null == definition)
      {
        try
        {
          definition = defGroup.Definitions.Create(
              defName, defType, visible);
        }
        catch (Exception)
        {
          definition = null;
        }
      }
      return definition;
```

```
    }
</C#>
```

3. Create shared parameter for doors



All we need in order to create a shared parameter for doors is the shared parameters file, a parameter definition group, the parameter definition itself and then binding it to the doors category.

First get the shared parameters file and make sure that our parameters group exists

```
<C#>
      // Get the current shared params definition file
      DefinitionFile sharedParamsFile = GetSharedParamsFile(app);
      if (null == sharedParamsFile)
      {
        message = "Error getting the shared params file.";
        return Result.Failed;
      }

      // Get or create the shared params group
      DefinitionGroup sharedParamsGroup = GetOrCreateSharedParamsGroup(
        sharedParamsFile, kSharedParamsGroupAPI);
      if (null == sharedParamsGroup)
      {
        message = "Error getting the shared params group.";
        return Result.Failed;
```

```
      }
</C#>
```

Then get back the category we need (doors) and create our shared parameter definition

```
<C#>
      Category cat =
          doc.Settings.Categories.get_Item(BuiltInCategory.OST_Doors);

      // Visibility of the new parameter:
      // Category.AllowsBoundParameters property indicates if a category can
      // have shared or project parameters. If it is false,
      // it may not be bound
      // to shared parameters using the BindingMap.
      // Please note that non-user-visible
      // parameters can still be bound to these categories.

      bool visible = cat.AllowsBoundParameters;

      // Get or create the shared params definition

      Definition fireRatingParamDef = GetOrCreateSharedParamsDefinition(
        sharedParamsGroup, ParameterType.Number,
        kSharedParamsDefFireRating, visible);

      if (null == fireRatingParamDef)
      {
        message = "Error in creating shared parameter.";
        return Result.Failed;
      }
</C#>
```

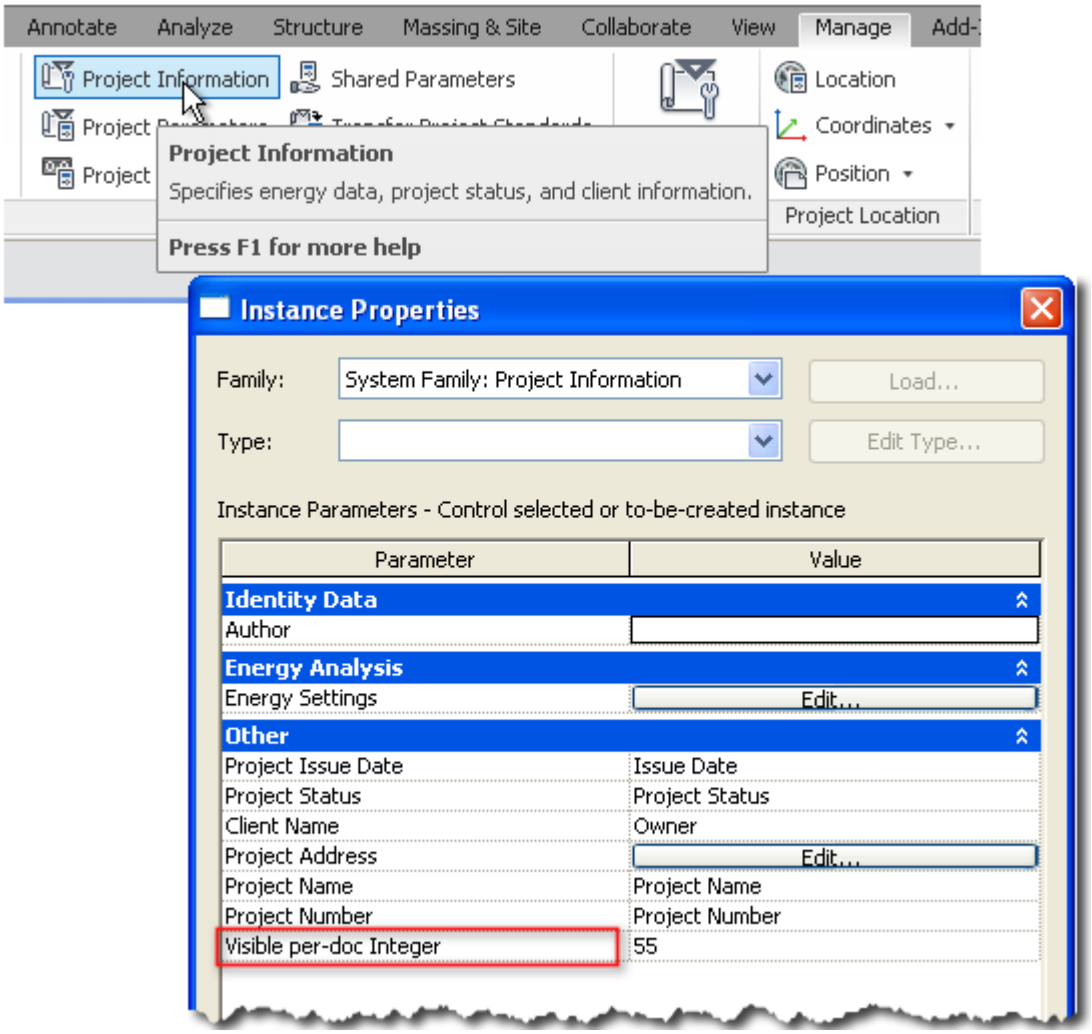
Once it's created we can bind it to the doors category and we're done.

```
<C#>
      CategorySet catSet = app.Create.NewCategorySet();
      try
      {
        catSet.Insert(cat);
      }
      catch (Exception)
      {
        message = string.Format(
          "Error adding '{0}' category to parameters binding set.",
          cat.Name);
        return Result.Failed;
      }

      // Bind the param
      try
      {
        Binding binding = app.Create.NewInstanceBinding(catSet);

        // We could check if already bound, but looks like Insert will
        // just ignore it in such case
```

```
        doc.ParameterBindings.Insert(fireRatingParamDef, binding);
      }
      catch (Exception ex)
      {
        message = ex.Message;
        return Result.Failed;
      }
</C#>
```

Now, when selecting a door in the user interface we should be able to see the parameter listed in the properties palette – as was shown above.

4. Create per document parameter



Now add a new command that creates a visible and an invisible per document parameter. By per document parameter we mean a parameter that is added to an element that has only a single instance in any project, and that is the Project Information element. The only difference in this exercise from the

previous one is that this time you need to bind the created parameter definition to the BuiltInCategory.OST_ProjectInformation category.

Also, find the Project Information element in the project using FilteredElementCollector or get it directly through property Document.ProjectInformation and then set its newly created parameter's value to something of your choice.

5. Summary

In this lab, we learned how to create shared parameters. We've learned how to:

- Create shared parameter group
- Create shared parameter definition
- Bind it to a specific category – e.g. doors