# Lab 5 – ExtensionDictionary and Named Objects Dictionary

In this lab, you will add an ExtenstionDictionary to selected entities. You will create an Xrecord that contains data and add this to the entities’ ExtensionDictionary. You will also add an Xrecord to the NamedObjectsDictionary. (NOD).

The steps for this lab are in two commands. The lab already has the code to add the commands and do a few other steps that we have already covered in previous labs. As you can see here the comments are the steps. You can copy this command to the cs file in your existing project (need to copy inside of an existing class) or open the Lab5 project that already has these command and the steps. Notice the two commands are “adddata” and “addDataToNOD”

[CommandMethod("addData")]

public void addData()

{

// get the editor object

Editor ed = Application.DocumentManager.MdiActiveDocument.Editor;

// pick entity to add data to!

PromptEntityResult getEntityResult = ed.GetEntity("Pick an entity to add an Extension Dictionary to : ");

// if all was ok

if ((getEntityResult.Status == PromptStatus.OK))

{

// now start a transaction

Transaction trans = ed.Document.Database.TransactionManager.StartTransaction();

try

{

// Start of Lab5

// Here we will add XData to a selected entity.

// 1. Declare an Entity variable named ent. Instantiate it using

// GetOBject of the Transaction created above. (trans) For the

// ObjectId parameter use the ObjectId property of the the

// PromptEntityResult created above.(getEntityResult) Open the entity for read.

// 2. Use an "if" statement and test the IsNull property of the

// ExtensionDictionary of the ent. If it is Null then we create

// the ExtensionDictionary.

// Note: Place the closing curly brace after step 4.

// 3. Upgrade the open of the entity. Because it does

// not have an extenstion dictionary and we want to add it

// the ent needs to be open for write.

// 4. Create the ExtensionDictionary by calling

// CreateExtensionDictionary of the entity.

// 5. Declare a variable as DBDictionary. Instantiate it by using the

// GetObject method of the Transaction created above. (trans). For the

// ObjectId parameter use the ExtensionDictionary property of the ent

// variable created in step 1. Open it for read

// 6. Check to see if the entry we are going to add to the dictionary is

// already there. Use the Contains property of the dictionary in an "if else

// statement.

// Note: Place the closing curly brace and the "else" after step 12. Place the

// closing curly brace of the "else" after after step 25

// 7. Declare an ObjectId variable named entryId and instantiate it

// using the GetAt method of the ExtenstionDictionary from step 5. Use

// "Mydata" for the entryName

// 8. If this line gets hit then data is already added

// Use the WriteMessage method of the Editor (ed) created above

// to print the data. For the string argument use something like this:

// "\nThis entity already has data..."

// 9. Now extract the Xrecord. Declare an Xrecord variable.

// 10. Instantiate the Xrecord variable using the

// GetObject method of the Transaction created above. (trans).

// For the ObjectId argument use the ObjectId created in step 7

// open for read.

// 11. Here print out the values in the Xrecord to the command line.

// Use a "foreach" statement. For the Element type use a TypedValue.

// (Use value for the name of the TypedValue) For the Group use the

// Data property of the Xrecord

// Note: Put the closing curly brace below step 12

// 12. Use the WriteMessage method of the Editor created above. (ed).

// for the string argument use something like this:

// "\n" + value.TypeCode.ToString() + " . " + value.Value.ToString()

// 13. If the code gets to here then the data entry does not exist

// upgrade the ExtensionDictionary created in step 5 to write by calling

// the UpgradeOpen() method

// 14. Create a new XRecord. Declare an Xrecord variable as a New Xrecord

// 15. Create the resbuf list. Declare a ResultBuffer variable. Instantiate it

// by creating a New ResultBuffer. For the ParamArray of TypeValue for the new

// ResultBuffer use the following:

//new TypedValue((int)DxfCode.Int16, 1),

//new TypedValue((int)DxfCode.Text, "MyStockData"),

//new TypedValue((int)DxfCode.Real, 51.9),

//new TypedValue((int)DxfCode.Real, 100.0),

//new TypedValue((int)DxfCode.Real, 320.6)

// 16. Add the ResultBuffer to the Xrecord using the Data

// property of the Xrecord. (make it equal the ResultBuffer

// from step 15)

// 17. Create the entry in the ExtensionDictionary. Use the SetAt

// method of the ExtensionDictionary from step 5. For the SearchKey

// argument use "MyData". For the DBObject argument use the Xrecord

// created in step 14.

// 18. Tell the transaction about the newly created Xrecord

// using the AddNewlyCreatedDBObject of the Transaction (trans)

// 19. Here we will populate the treeview control with the new data

// Use an "if" statement to check to see if the

// palette (myPalette created in Lab 4) is not equal to null.

// (If not then it will crash)

// Note: Put the closing curly brace after step 25

// 20. Create a foreach statement. Use node for the element name and

// the type is System.Windows.Forms.Treenode. The group paramater is the Nodes in the

// TreeView. (myPalette.treeView1.Nodes)

// Note: put the closing curly brace after step 25

// 21. Use an "if" statement. Test to see if the node Tag is the ObjectId

// of the ent from step 1. Use the ObjectId. (ent.ObjectId.ToString)

// Note: put the closing curly brace after step 25

// 22. Now add the new data to the treenode. Declare a variable as a

// System.Windows.Forms.Treenode. (name it something like childNode).

// Instantiate it by making it equal to the return of calling the Add

// method of the Nodes collection of the node from the loop.

// (node.Nodes.Add) For the string argument use "Extension Dictionary"

// 23. Now add the data. Create a foreach statement. Use value for the element

// name and the type is TypedValue. Use the Data property of the Xrecord created in

// step 14 for the group.

// Note: put the closing curly brace after step 24

// 24. Add the TypeValue from the For Each loop to the

// TreeNode created in step 22. Use the Add method of the

// Nodes Collection. (childNode.Nodes.Add) For the string

// argument use the TypeValue from the loop. (value.ToString)

// 25. Exit the for loop (all done - break out of the loop)

// End of Lab 5

// all ok, commit it

trans.Commit();

}

catch (Exception ex)

{

// a problem occured, lets print it

ed.WriteMessage("a problem occured because " + ex.Message);

}

finally

{

// whatever happens we must dispose the transaction

trans.Dispose();

}

}

}

[CommandMethod("addDataToNOD")]

public void addDataToNOD()

{

// get the editor object

Editor ed = Application.DocumentManager.MdiActiveDocument.Editor;

// pick entity to add data to!

Transaction trans = ed.Document.Database.TransactionManager.StartTransaction();

try

{

// 26. Here we will add our data to the Named Objects Dictionary.(NOD)

// Declare a variable as a DBDictionary. (name it nod). Instantiate it

// by making it equal to the return of the GetObject method of the

// Transaction created above. (trans). For the ObjectId argument use the

// NamedObjectsDictionaryId property of the current Database:

// (ed.Document.Database.NamedObjectsDictionaryId) The Editor (ed) was

// instantiated above. Open it for read.

// 27. Check to see if the entry we are going to add to the NOD is

// already there. Use the Contains property of the dictionary in an "if Else"

// statement.

// Note: put the closing curly brace and the else after step 33. Put the closing

// curley brace for the else after step 39

// 28. Declare an ObjectId variable named entryId. Instantiate it by making

// it equal to the return of the GetAt method of the NOD (DBDictionary)

// from step 26. For the EntryName agrument use "MyData"

// 29. If we are here, then the Name Object Dictionary already has our data

// Use the WriteMessage method of the editor. Use this for the Message argument

// "\n" + "This entity already has data..."

// 30. Get the the Xrecord from the NOD. Declare a variable as a new Xrecord

// 31. USe the Transaction (trans) and use the GetObject method to

// get the the Xrecord from the NOD. For the ObjectId argument use the

// ObjectId from step 28. Open the Xrecord for read

// 32. Print out the values of the Xrecord to the command line. Use

// a "foreach" statement. Use value for the element

// name and the type is TypedValue. Use the Data property of the Xrecord from

// step 31 for the group argument

// Note: put the closing curly brace after step 33

// 33. Use the WriteMessage method of the editor. Use this as the message:

// "\n" + value.TypeCode.ToString() + " . " + value.Value.ToString()

// 34. Our data is not in the Named Objects Dictionary so need to add it

// upgrade the status of the NOD variable from step 26 to write status

// 35. Declare a varable as a new Xrecord.

// 36. Create the resbuf list. Declare a ResultBuffer variable. Instantiate it

// by creating a New ResultBuffer. For the ParamArray of TypeValue for the new

// ResultBuffer use the following:

// new TypedValue((int)DxfCode.Int16, 1),

//new TypedValue((int)DxfCode.Text, "MyCompanyDefaultSettings"),

//new TypedValue((int)DxfCode.Real, 51.9),

//new TypedValue((int)DxfCode.Real, 100.0),

//new TypedValue((int)DxfCode.Real, 320.6)

// 37. Add the ResultBuffer to the Xrecord using the Data

// property of the Xrecord. (make it equal the ResultBuffer

// from step 36)

// 38. Create the entry in the ExtensionDictionary. Use the SetAt

// method of the Named Objects Dictionary from step 26. For the SearchKey

// argument use "MyData". For the DBObject argument use the Xrecord

// created in step 35.

// 39. Tell the transaction about the newly created Xrecord

// using the AddNewlyCreatedDBObject of the Transaction (trans)

// all ok, commit it

trans.Commit();

}

catch (Exception ex)

{

// a problem occurred, lets print it

ed.WriteMessage("a problem occurred because " + ex.Message);

}

finally

{

// whatever happens we must dispose the transaction

trans.Dispose();

}

}