# SqlDataGridViewEditor.translationPlugin

## DataGridView (to be changed)

public class DataGridViewPlugin : IPlugin

{

private ControlTemplate controlTemplate;

private String name = String.Empty;

public DataGridViewPlugin(String name)

{

this.name = name;

controlTemplate = new ControlTemplate(this.Name(),

new List<string>() { "Load Data" },

new DataGridViewUserControl());

}

public String Name()

{

return this.name;

}

public ControlTemplate PluginControls()

{

return controlTemplate;

}

}

## DataGridViewUserControl (Form)

namespace SqlDataGridViewEditor.TranscriptPlugin

{

public partial class DataGridViewUserControl : UserControlWithCallBack

{

public DataGridViewUserControl()

{

InitializeComponent();

base.CallBack += DataGridViewUserControl\_CallBack;

}

void DataGridViewUserControl\_CallBack(object sender, EventArgs<string> e)

{

if (e.Value == "Load Data")

{

DataTable testData = MockData.GenerateDataTable<Person>(50);

dataGridViewTest.DataSource = testData;

lblDescription.Visible = true;

dataGridViewTest.Visible = true;

}

}

}

}

## MockData.cs

public class MockData

{

public static DataTable GenerateDataTable<T>(int rows)

{

var datatable = new DataTable(typeof(T).Name);

typeof(T).GetProperties().ToList().ForEach(x => datatable.Columns.Add(x.Name));

Builder<T>.CreateListOfSize(rows).Build().ToList()

.ForEach(x => datatable.LoadDataRow(x.GetType().GetProperties()

.Select(y => y.GetValue(x, null)).ToArray(), true));

return datatable;

}

}

# SqlDataGridViewEditor (Main program)

## Plugin.cs

internal static class Plugins

{

static IUnityContainer? container = null;

static internal String pluginFilePath = String.Empty;

static internal MenuStrip Load\_Plugins()

{

MenuStrip plugInMenus = new MenuStrip();

pluginFilePath =   
 Directory.GetParent(System.IO.Directory.GetCurrentDirectory()).Parent.Parent.FullName + @"\PluginsToConsume\";

container = new UnityContainer();

string[] files = Directory.GetFiles(pluginFilePath, "\*.dll");

Int32 pluginCount = 1;

foreach (String file in files)

{

Assembly assembly = Assembly.LoadFrom(file);

foreach (Type T in assembly.GetTypes())

{

foreach (Type iface in T.GetInterfaces())

{

if (iface == typeof(IPlugin))

{

IPlugin pluginInstance = (IPlugin)Activator.CreateInstance(T, new[] { "Live Plugin " + pluginCount++ });

container.RegisterInstance<IPlugin>(pluginInstance.Name(), pluginInstance);

}

}

}

}

// At this point the unity container has all the plugin data loaded onto it.

if (container != null)

{

var loadedPlugins = container.ResolveAll<IPlugin>();

if (loadedPlugins.Count() > 0) { }

foreach (var loadedPlugin in loadedPlugins)

{

plugInMenus.Items.Add(loadedPlugin.PluginControls().MenuStripItemContainer);

// TabPage tabPage = new TabPage(loadedPlugin.Name());

// tabPage.Controls.Add(loadedPlugin.PluginControls().UserControlContainer);

// hostTabControl.TabPages.Add(tabPage);

}

}

return plugInMenus;

}

}

## Ddd

public partial class InheritedUserControl : UserControlWithCallBack

{

public InheritedUserControl()

{

InitializeComponent();

base.CallBack += InheritedUserControl\_CallBack;

}

void InheritedUserControl\_CallBack(object sender, EventArgs<string> e)

{

MessageBox.Show("Message From ToolStrip, Received in Inherited User Control : " + e.Value);

}

private void button1\_Click(object sender, EventArgs e)

{

MessageBox.Show("Message From Inherited User Control");

}

}