# Module 5 Lab A

# Creating a Custom Plug-In

##### Objective: Create a custom Plug-in that will attempt to resolve missing address information associated with a contact record.

**Step1: Creating a new Dynamics Solution and Create a Visual Studio project to support plugin development.**

|  |  |
| --- | --- |
| 1 | Create a new Solution in Dynamics. Name it Modules 5 |
| 2 | Open a new instance of Visual Studio 2015 Community edition as administrator.  Create a new Visual Studio project in the Module 5 LAB A Starter folder. Name the solution AddressResolver and Select the Dynamics 365 Plug-in Library as the template.  You will be prompted to connect to a Dynamics 365 system and then select the solution (Select Module 5).  Build the solution and verify that it complies without error |

|  |  |
| --- | --- |
| 3 | Inside the PluginBase class there is the Execute method that will be called by Dynamics 365 in the processing pipeline.  This in turn creates references to services that allow tracing info to be written and information about the execution context to be accessed along with the ability to create a client to the Organization Service.  The Exexute method calls a named ExecuteCrmPlugin which is virtual which we can override in a derived class. (as we will) |

**Step 2: Create the AddressResolver plugin**

|  |  |
| --- | --- |
| 1 | Add a class to the project created in Step 1, named AddressFromZipCodePlugin and change the access modifier to public and let it inherit from PluginBase. |
| 2 | You should see a warning that needs to be fixed. Add the following constructor to the AddressFromZipCodePlugin class  public AddressFromZipCodePlugin(string unsecure, string secure) :  base(typeof(AddressFromZipCodePlugin))  {  }  The parameters to the constructor can be used to pass configuration data to the plug-in. In lab B we will use this to pass the URL of the web service to the plug-in |
| 3 | Add the following using statements to the existing ones in the class.  using Microsoft.Xrm.Sdk;  using Microsoft.Xrm.Sdk.Query;  using System.IO;  using System.Net; |

|  |  |
| --- | --- |
| 4 | In the AddressFromZipCodePlugin class override the ExecuteCrmPlugin method.  The localcontext parameter reference can be used to access information about the new entity (in our case a contact).  Assign the localcontext. PluginExecutionContext to a variable named context as the first line of the Method. |
| 5 | Immediately below the variable declaration add code that uses the variable to check that the plug-is executing based on a message that is associated with an Entity and that the entity is a “contact”.  If not get the method to return immediately. |
| 6 | Add an else statement to the nested if statement (second if statement) that calls a method named resolveAddressIfNecessary passing in the variables named contact and localcontext as parameters.  Get visual studio to generate the missing method in the usual way. |
| 7 | Remove the throw new NotImplementedException(); statement in the method created by the IDE.  The resolveAddressIfNecessary method is passed the new contact entity.  We will use the information passed to identify whether or not it’s necessary to make an outbound call to the service that will resolve the zipcode to an address. |
| 8 | Within the resolveAddressIfNecessary method add code to check if the Postal Code has a value but the city is empty and if is the case assign the value to the postal code to a string variable named strZipCode, as follows. |

|  |  |
| --- | --- |
| 9 | Open a Web Browser and navigate to http://webservicex.net/uszip.asmx This is the service we will be calling from the plugin. You should see the help page that allows you to view this web service’s methods and test them. |
| 13 | The web service supports simple http get requests using querystring parameters and that’s how we’ll call it |
| 14 | Click on the GetInfoByZip method and test it by entering a valid us zip code like 90210 into the input box.  Note that the CITY and STATE are returned for the request. |
| 15 | We will add code that uses low level HttpRequest and HttpResonse classes to call the web service. |
| 16 | Go back to the resolveAddressIfNecessary method and immediately below the line that declares and initialises the strZipCode variable.  Create a try catch block (Catching exceptions of type Exception).  Within the try block, declare a variable named httpClient using the var keyword.  Assign to it the result of calling the create static method of the HttpWebRequest class passing a vaild URL. Following that call the GetResponse method of the client assigning it to a variable named response. See below |
| 17 | Immediately below the variable named response add the following code statements.  Note that The content returned by the web service is in xml format and the code above searches for the city and state elements and extracts the values from these elements.  StreamReader reader = new StreamReader(response.GetResponseStream());  string xmlData = reader.ReadToEnd();  int beginLocation = xmlData.IndexOf("<CITY>");  int endLocation = xmlData.IndexOf("</CITY>");  string city = xmlData.Substring(beginLocation + 6, endLocation - beginLocation - 6);  beginLocation = xmlData.IndexOf("<STATE>");  endLocation = xmlData.IndexOf("</STATE>");    string state = xmlData.Substring(beginLocation + 7, endLocation - beginLocation - 7); |
| 18 | Now update the contact entity referenced by the variable contact so that the City and State information returned by the web service updates the address1\_city and address1\_stateorprovince attributes.  The localcontext object exposes an OrganzationService property.  Call the update method of the OrganzationService passing the variable named contact as a parameter. This will update the contact with the address information returned with the webservice call. |
| 19 | In the catch bloc call the localContext.TracingService Trace method to log information about the exception such as the Message etc. |
| 20 | Sign the assembly using the snk file created previously and then build the solution and fix any errors. |

**Step 4: Deploying and registering the AddressResolver plugin**

|  |  |
| --- | --- |
| 1 | Open the PluginReistration tool from the tools folder in the SDK and connect to your Dynamics 365 Organisation |
| 2 | Click on the register button and navigate to the dll generated when the project was compiled. The click on the Select all button followed by the register button at the bottom of the dialog. |
| 3 | We will now associate the Plug-In with the contact create operation. Right click on your custom plugin and select register new step. |
| 4 | Associate execution of the plugin with the contact create message and mark it as a Post –Operation (after the contact record has be created) and asynchronous. |
| 5 | Test the plug-in by creating a contact record that has a us zip code but not the city and after its creation open it up to see if the plug-in has filled in the missing elements.  If the plug-in fails, then use the debugging techniques explored in the previous module to resolve the issues. Also ask the instructor for help |