# Module 2 Lab B Tooling Connector

Time 10 Minutes

##### Objective: Convert the previous solution to use the new Tooling Connector API

##### 

##### Actions

**Step 1: Copy Module 2 Lab A solution to the starter folder for Module 2 Lab B**

|  |  |
| --- | --- |
| 1 | Copy and paste your finished solution to lab A into the starter folder for lab B ( if you didn’t get to finish lab A use the solution provided in Lab the A END folder |
| 2 | Open a new instance of Visual Studio 2015 as administrator and open the copied solution in the Module 2 LAB B starter folder |
| 3 | ~~Add a references to the existing project references to the Microsoft.Xrm.Tooling.Connector.dll located in the Bin directory of the SDK C:\Program Files\Microsoft SDKs\SDK\Bin~~  ~~and a reference to the System.Configuration.dll~~  Add a reference to the nuget package Microsoft.XrmSdk.XrmTooling.Coreassembly |
| 4 | Modify the app.config file so that it includes a valid connection string to your Dynamics 365 online Tenancy.  e.g.  <connectionStrings>  <add name="CrmOnline" connectionString="Url=https://qajuly17.crm11.dynamics.com;Username=paul@qajuly17.onmicrosoft.com;Password=QaJuly17;authtype=Office365" />  </connectionStrings>  Note: replace the username and password and the url so that they match ones correct for your tenancy. |
| 5 | Inside Form1 Class, declare a private parameterless function named createOrgServiceClient that returns an object of type CrmServiceClient    Now fix the issues related to the missing using statement by clicking on the red line under the CrmServiceClient datatype and the hit the CTRL+. Key combination allowing it to add the appropriate using statement to the Form1.cs file    There will still be red warnings related to the function not returning an object of type CrmServiceClient. Which will resolved soon. |
| 6 | Add to the existing using statements at the top of the file, a using statement to the System.Configuration namespace.  Add a statement to the createOrgServiceClient function that reads the connection string and stores it in a string variable named cnString;.  Below the statement you’ve just added create a new instance of a CrmServiceClient passing the cnString variable to the constructor and assign this to the return value of the function . You should end up with the function shown below  private CrmServiceClient createOrgServiceClient()  {  string cnString = ConfigurationManager.ConnectionStrings["CrmOnline"].ConnectionString;  return new CrmServiceClient(cnString);  } |
|  | Modify the button1\_click event handler. Update the using statement and add calls to create the CrmServiceClient and assign it to a variable of that type named cmrServiceClient.  Note changes have been circled in red. |
|  | Run the app and check that it continues to function correctly  Conclusion you can use the Tooling Connector library to create a proxy object that will communicate with the Dynamics 365 OrganisationalServices Web Service. |