# Module 2 Lab E

# Using FetchXML

Time 10 Minutes

##### Objective: Use the FetchXML to return the total number of account records in our system.

**Step 1: Modify the Visual Studio project completed in Module 2 Lab D**

|  |  |
| --- | --- |
| **1** | Run Visual Studio 2015 Community Edition Open the completed solution for Lab D |

|  |  |
| --- | --- |
| **2** | Edit the Accounts controller, add a public method, named GetAccountsCount that return a string. This method will be used to return the number of Account records in our system. In order to get aggregate data we need to execute a FetchXML request. For the moment get it to return an empty string.  public int GetAccountsCount()  {  return 0;  } |
| **3** | Within the GetAccountCount function declare a string variable named numberOfAccountsFetchXml and assign it the string as below  string numberOfAccountsFetchXml = @"  <fetch distinct='false' mapping='logical' aggregate='true'>  <entity name='account'>  <attribute name='name' alias='account\_count' aggregate='count'/>  </entity>  </fetch>";  This FetchXML query when executed will retrieve an aggregate count on the Account entity and allow the result to be accessed via the alias account\_count. |
| **4** | Below the string variable declare a new instance of the CrmServiceClient initialising it with the connection string for your Dynamics 365 tenancy and wrapping the calling the usual using statement.  using (var crmSvc = new CrmServiceClient(cnString))  {  } |
| **5** | Within the using statement declare a variable named numberOfAccounts of type EntityCollection and assign to it the result of calling the RetreiveMultiple method of the crmSvc object.  Pass as a parameter to the RetreiveMultiple method a new instance of a FetchExpression that has been initialised with xml stored in the string variable numberOfAccountsFetchXml.  EntityCollection numberOfAccounts = crmSvc.RetrieveMultiple(new FetchExpression(numberOfAccountsFetchXml)); |
| **6** | Following create an if statement. It the numberOfAccounts is not null then use the first item in the entity collection to extract the value via it’s alias “account\_count” and assign it to an integer variable. Return the int result as a string. In the else block return the string “Error”  if (numberOfAccounts != null)  {  int number = (int)((AliasedValue)numberOfAccounts[0]["account\_count"]).Value;  return number;  }  else  {  return 0;  } |
| **7** | Add the following line to the Paging method in the account controller. |
| **8** | Modify the paging.cshtml file add the <h3> HTML element below the div tag with the class named navbar-brand and call the action on the controller linked to the fetchXml request. This will display the total number of account records available. In addition to this add a link that will call the Paging action passing the value of ViewBag.lastPage to the id parameter of the Paging action method. |
| **9** | Test that the Application now allows a navigation to the last page of account records and displays the total number of Account records. |