# Logging Entity Framework Core operations

Logging can assist with queries that do not run as expected which can range from improper joins to values passed to where clauses.

Note when working with SQL-Server databases SSMS provides profiling tools to monitor queries. At OED this can be difficult as there are a good deal of actions happening.

The following lambda statement utilizes an extension [TagWith](https://docs.microsoft.com/en-us/dotnet/api/microsoft.entityframeworkcore.entityframeworkqueryableextensions.tagwith?view=efcore-5.0) which is known as a query tag.

Query tags help correlate LINQ queries in code with generated SQL queries captured in logs. You annotate a LINQ query using the new TagWith() method.

public static async Task<List<CustomerItem>> GetCustomersAsync()

{

    var currentExecutable = Process.GetCurrentProcess().MainModule.FileName;

    return await Task.Run(async () =>

    {

        using (var context = new NorthwindContext())

        {

            return await context.Customers.AsNoTracking()

                .Include(customer => customer.Contact)

                .ThenInclude(contact => contact.ContactDevices)

                .ThenInclude(contactDevices => contactDevices.PhoneTypeIdentifierNavigation)

                .Include(customer => customer.ContactTypeIdentifierNavigation)

                .Include(customer => customer.CountryIdentifierNavigation)

                .Select(customer => new CustomerItem()

                {

                    CustomerIdentifier = customer.CustomerIdentifier,

                    CompanyName = customer.CompanyName,

                    ContactId = customer.Contact.ContactId,

                    Street = customer.Street,

                    City = customer.City,

                    PostalCode = customer.PostalCode,

                    CountryIdentifier = customer.CountryIdentifier,

                    Phone = customer.Phone,

                    ContactTypeIdentifier = customer.ContactTypeIdentifier,

                    Country = customer.CountryIdentifierNavigation.Name,

                    FirstName = customer.Contact.FirstName,

                    LastName = customer.Contact.LastName,

                    ContactTitle = customer.ContactTypeIdentifierNavigation.ContactTitle,

                    OfficePhoneNumber = customer.Contact.ContactDevices.FirstOrDefault(contactDevices =>

                        contactDevices.PhoneTypeIdentifier == 3).PhoneNumber

                })

                .TagWith($"App name: {currentExecutable}")

                .TagWith($"From: {nameof(CustomersTestOperations)}.{nameof(GetCustomersAsync)}")

                .TagWith("Parameters: None")

                .ToListAsync();

        }

    });

}

Starting with Entity Framework Core 5 logging has become extremely easy while earlier versions required more work.

## First option is using

EntityFrameworkQueryableExtensions.ToQueryString(IQueryable) Method

Consider the following code from one of the unit test methods created earlier.

using var context = new NorthwindContext();

var customers = context.Customers.ToList();

To peek under the covers

using var context = new NorthwindContext();

var sqlFromQuery = context.Customers.ToQueryString();

Debug.WriteLine(sqlFromQuery);

* Requires: using Microsoft.EntityFrameworkCore;

Produces

SELECT [c].[CustomerIdentifier], [c].[City], [c].[CompanyName], [c].[ContactId], [c].[ContactTypeIdentifier], [c].[CountryIdentifier], [c].[Fax], [c].[ModifiedDate], [c].[Phone], [c].[PostalCode], [c].[Region], [c].[Street] FROM [Customers] AS [c]

## Second method

Logging to text file (Karen will do a demonstration)

There is an option to create a new log file or append to an existing log file.

Suppose we need to look at a lambda or LINQ query. In the test method below we are passing a number to return a customer.

In a complex code set it may be difficult to know what the number is so we use logging.

[TestMethod]

[TestTraits(Trait.EfCoreCustomersSelect)]

public void SingleCustomerByIdentifierGood()

{

    int customerIdentifier = 1;

    CustomerEntity customer = CustomersOperations.CustomerByIdentifier(customerIdentifier);

    Assert.IsNotNull(customer);

    Assert.IsTrue(customer.CompanyName == "Alfreds Futterkiste");

}

Here is an extract from running the test method above. The number shows up in the highlighted text.

(Microsoft.EntityFrameworkCore.Database.Command)

Executed DbCommand (59ms) [Parameters=[@\_\_identifier\_0='1'], CommandType='Text', CommandTimeout='30']

SELECT TOP(1) [c].[CustomerIdentifier], [c].[CompanyName], [c].[Street], [c].[City], [c].[PostalCode], [c].[ContactTypeIdentifier], [c0].[ContactTitle], [c].[ContactId], [c1].[Name], [c2].[FirstName], [c2].[LastName], [c2].[ContactId], [c2].[ContactTypeIdentifier], [c].[CountryIdentifier], [c1].[CountryIdentifier], [c].[ModifiedDate]

FROM [Customers] AS [c]

LEFT JOIN [ContactType] AS [c0] ON [c].[ContactTypeIdentifier] = [c0].[ContactTypeIdentifier]

LEFT JOIN [Countries] AS [c1] ON [c].[CountryIdentifier] = [c1].[CountryIdentifier]

LEFT JOIN [Contacts] AS [c2] ON [c].[ContactId] = [c2].[ContactId]

WHERE [c].[CustomerIdentifier] = @\_\_identifier\_0

## Third method

Using the same configuration for logging to a file, we can write to Visual Studio’s output window for application code or in the output window for a unit test method.

## More?

There are other more advance options also but we are not getting into them at this time.

# ASP.NET Core

Offers configuring logging from a json file such as shown below. Also note the connection string is stored in the same file.

{

  "Logging": {

    "Debug": {

      "LogLevel": {

        "LearnAspCore.Controllers.HomeController:Critical:": "Warning",

        "Default": "Warning",

        "Microsoft": "Information"

      }

    },

    "LogLevel": {

      "Default": "Warning",

      "Microsoft": "Information"

    }

  },

  "AllowedHosts": "\*",

  "SomeIncreptedKey": "\*&\*^%)hjaasdlaj\*&(",

  "ConnectionStrings": {

    "StudentDBString": "Data Source=.\\SQLEXPRESS;Integrated Security=True;database=NorthWind2020"

  }

}