Contents

The Query Explorer is a simple application, and utility, which you can use to learn LINQ.

- **Platform:** WinForms
- **Language:** C#, VB
- **Download:** Query Explorer (WinForms)

Problem

If you're new to LINQ, or not familiar with some of its more arcane syntax, a simple utility showing sample queries for many LINQ operators, and the query results, can be a handy tool.

Solution

The *Query Explorer* is an adaptation of a tool first distributed by Microsoft with the Entity Framework v1 called Entity Framework Query Samples. These samples were in turn based on the MSDN "101 LINQ Samples".

The DevForce version uses the same Microsoft sample database (NorthwindEF) and a "DevForce-enabled" model. The primary difference is that the DevForce samples use an EntityManager, and also illustrate other DevForce features such as control over query execution and offline support.

In DevForce a LINQ query may be executed either synchronously or asynchronously, although only synchronous query execution is shown in the utility.

Using the tool you can select a query and a QueryStrategy and then run the query to see the results. It also allows you to start a TraceViewer to see tracing activity while queries run.

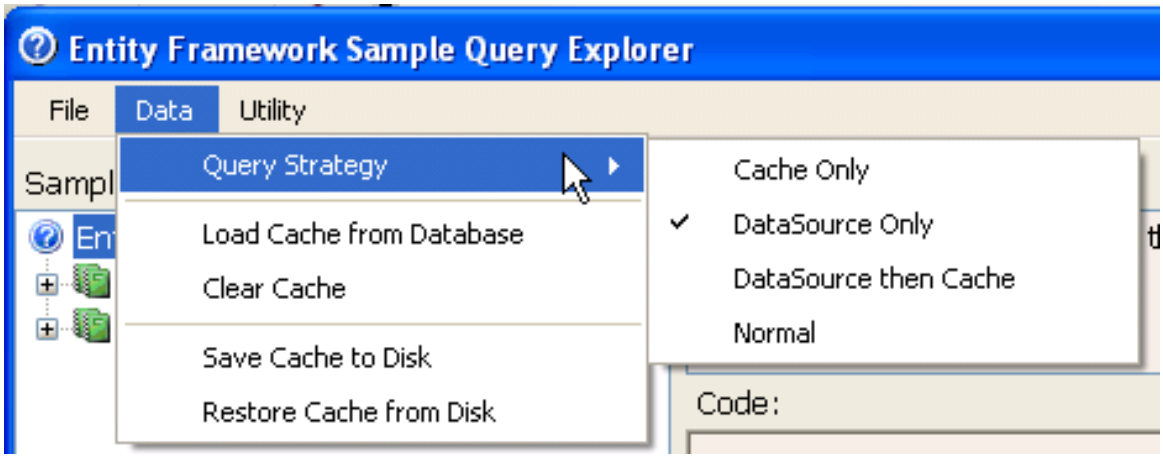
This is not a sample for how to build a WinForms application with DevForce.

What the utility does

- Run a variety of queries in connected and disconnected mode
- Save the contents of the EntityManager cache to a local file
- Restore the contents of that local file and run in disconnected mode

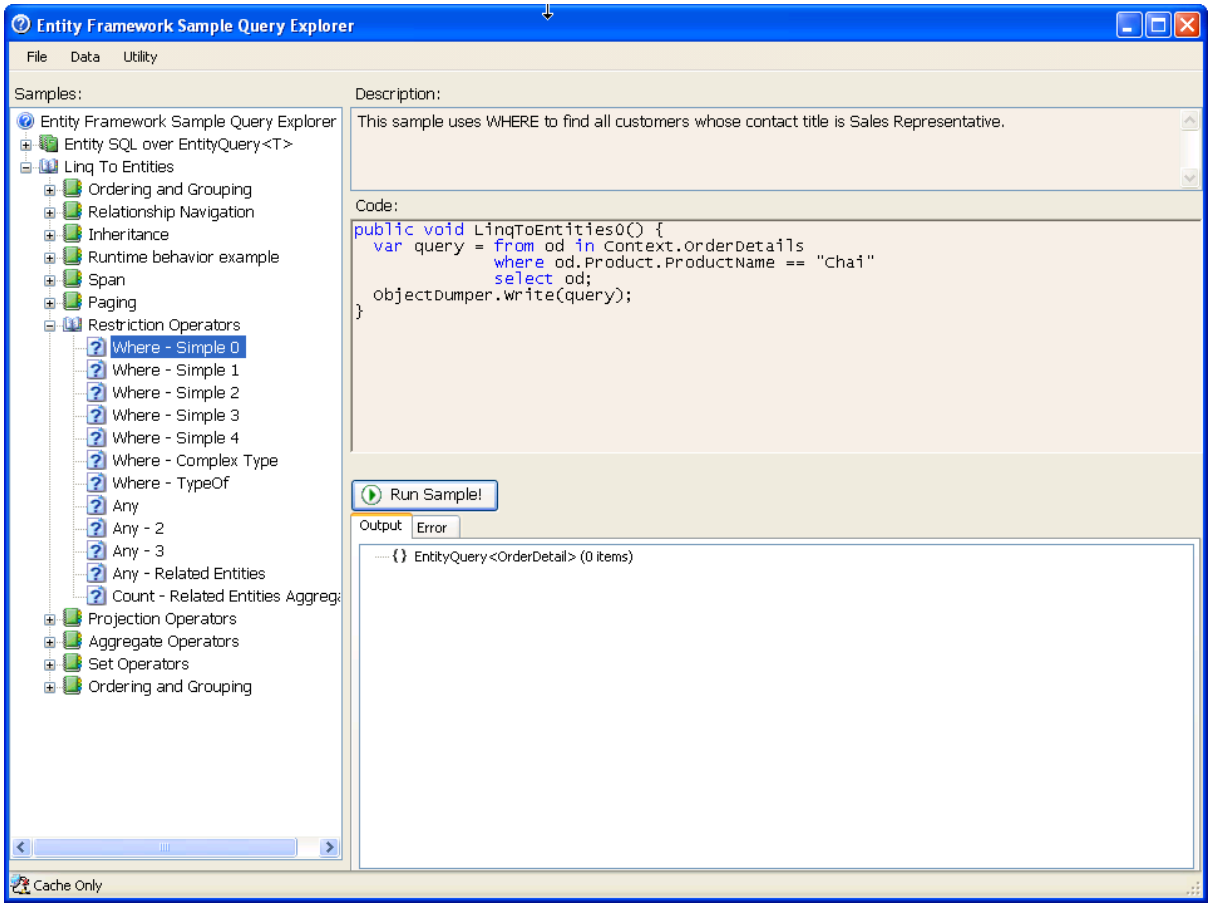
Here are a few things to try:

1. Select Data / Query Strategy from the main menu:

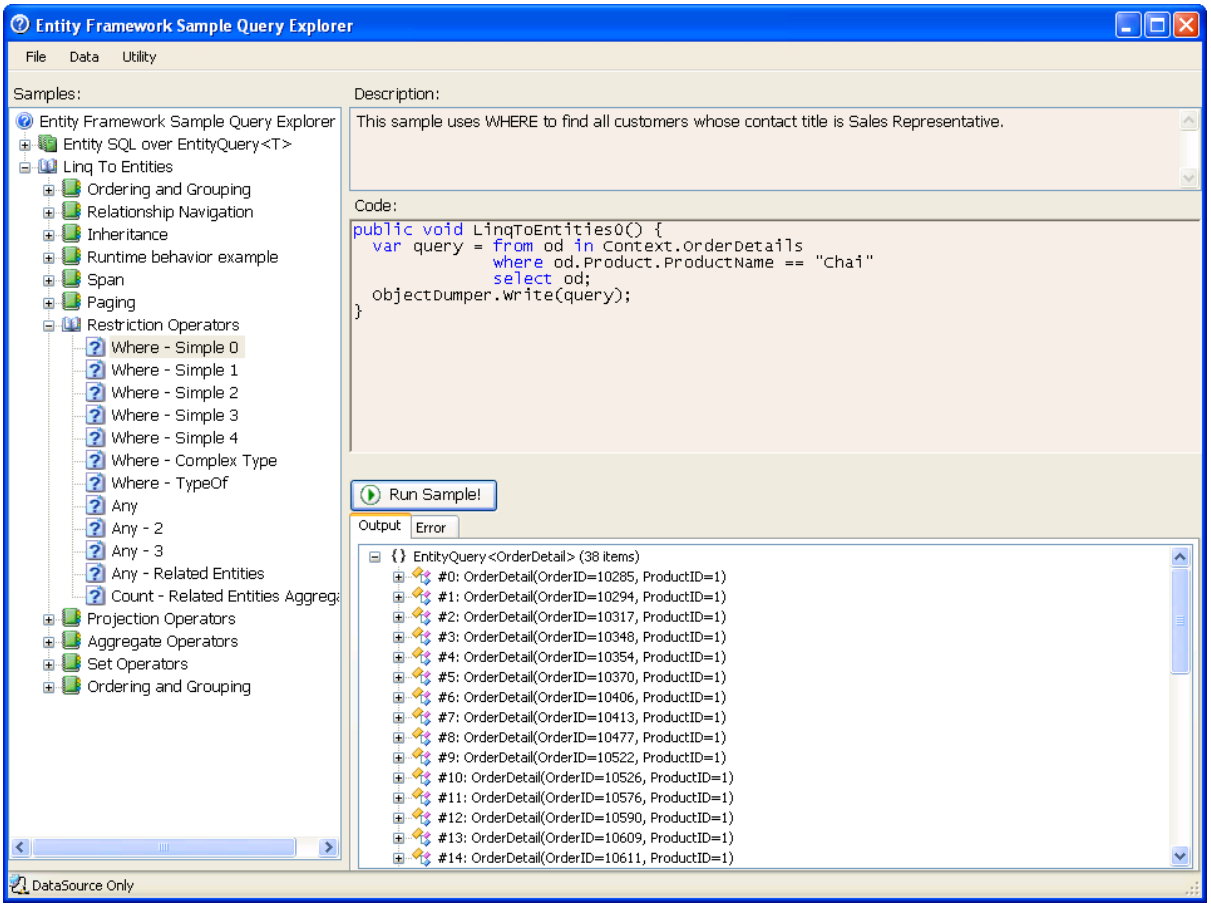


Note that the default QueryStrategy is DataSource Only, meaning that any selected query will be run against the backend datasource. But you can change the strategy to any of the others supported by DevForce to explore the action of the other QueryStrategies. (You can learn more about DevForce QueryStrategies in the Developers Guide and in the Data Retrieval tutorials.)

1. Change the QueryStrategy to CacheOnly and run some queries in the Linq to Entities \ Restriction Operators section. Note that you don't get much in the way of results, since the app has never been to the data source and so has nothing in the cache.



1. Change the QueryStrategy back to DataSourceOnly and run some queries. You should get more interesting results now!



1. Launch the DevForce TraceViewer from the Utility menu. Run some more queries and note the activity in the TraceViewer form.

1. Change the QueryStrategy back to CacheOnly and run a couple more queries. Note that absence of activity in the TraceViewer, indicating that no trip was made to the EntityServer. Nevertheless, many of the queries will return results at this point, because your previous queries loaded quite a bit of data in the local cache.

Some queries that returned results with the QueryStrategy of DataSourceOnly may not return results with the QueryStrategy of CacheOnly, even after having been run the first way. That occurs with some queries that must examine objects of a type different than the return type. See the topic of “Query Inversion” in the Developers Guide for details.

1. Clear the local cache by selecting Data / Clear Cache from the main menu. Now you will see neither server activity nor meaninful results.

1. Set the QueryStrategy back to DataSourceOnly; then select Data / Load Cache from the menu. This runs the following span query:

```
var query = context.Customers
  .Include("Orders")
  .Include("Orders.OrderDetails")
  .Include("Orders.OrderDetails.Product")
  .Include("Orders.OrderDetails.Product.Supplier");
query.QueryStrategy = QueryStrategy.DataSourceOnly;
query.ToList();
```

```
Dim query = context.Customers _
  .Include("Orders") _
  .Include("Orders.OrderDetails") _
  .Include("Orders.OrderDetails.Product") _
  .Include("Orders.OrderDetails.Product.Supplier")
query.QueryStrategy = QueryStrategy.DataSourceOnly
query.ToList()
```

This query loads into the cache all Customers from the data source, along with their associated Orders; the OrderDetails associated with those Orders; the Products referenced in the OrderDetails; and the Suppliers for the Products. Of course, in a real enterprise app, that would be a great deal of data to bring down to the client; but just adding a .Where() on the

context.Customers part of the query to limit yourself to one or a few Customers would make it a very realistic operation to perform. Having loaded all the data needed for the Customer or Customers on which you were working, your data maintenance operations would zip along at local bus speed!

1. Save the contents of the cache to a local disk file by selecting Data / Save Cache to Disk from the menu. (This app saves the data to a hard-coded location in the root of the C drive; feel free to change that if you wish.)
1. Shut down and re-start the app. Imagine you boarded a plane to Rio in the meantime, and are now flying over the Caribbean. Set the QueryStrategy to CacheOnly so that your queries will depend entirely on data loaded to the cache from some source other than the database (to which you would be unable to connect. (If you wish, test this by running a few queries now, and noting the absence of returned data.) Now select Data / Restore Cache from Disk to reload the data you saved to disk in the previous step.
1. Try running some queries. (Take your pick.) Almost all of them reference entities of the types that our span query loaded into the cache, so we have that data, and the queries work. As exceptions, see the “Where – Complex Type” and “Where – TypeOf” queries in the Linq to Entities / Restriction Operations section. These queries attempt to retrieve Employees, and since we didn’t have any Employees in the cache file we saved, we don’t have any now to work with.

Prerequisites

This solution uses the NorthwindEF database originally distributed by Microsoft. The NorthwindEF database is provided in a zip file in the Data folder of this learning unit. Unzip the component files and attach them to your instance of SQL Server. Be sure to check the *connectionStrings* in the app.config file too.