Linq To SQL: Sort Query by Arbitrary Property(Column) Name

Ask Question

11

I have a larger/more complex problem, but for simplicity sake, let us consider the following:

Let us say that I have table in the **SQL DataBase** called **Product**, having two columns, **ID** *(int, primary key)* and **Name** *(varchar/string)*. I also have a simple **LINQ DataContext**.

I have a query constructed and handed to “my” function. Let us say it is something like: *(though it may be a bit more complex)*

```
IQueryable<Product> query = from p in db.Products select p;
```

Once my method gets this query, passed in as a parameter, it has to change the sort order e.g.

```
IQueryable<Product> sortedQuery = query.OrderBy(x => x.Name);
```

I would like to make this more generic i.e. to specify the field to sort on. Normally, I can do a `switch` statement that takes a string. However I would like to know if there is a way to pass the parameter directly. I intend to extend this to other Database tables, so these `switch` statements would get tedious.

I was trying something like:

```
IQueryable<Product> sortedQuery = query.OrderBy(x =>
(typeof(Product)).GetProperty("Name"));
```

But this does not work. I also want to ensure that the LINQ to SQL is maintained i.e. the sort to be done on the SQL Server. Hence if I debug, I should get a SQL query from this LINQ query.

Thank you in advance for your help.

c# linq c#-4.0 linq-to-sql

asked Mar 13 '13 at 14:32

O.O.
721 5 17 29

Thanks to @Viper, I think I was really looking for **Dynamic LINQ** here. Using that I would simply have `IQueryable<Product> sortedQuery = query.OrderBy("Name asc");` which is what I was looking for. – O.O. Mar 13 '13 at 15:42

Without getting too excited and looking at the green tick solution, you have described my exact problem with perfection. I was trying similar solutions to you as well. Getting excited about scrolling down! – ozzy432836 Dec 16 '15 at 11:08

3 Answers

15

You could use `Dynamic Linq` for this purpose.

See here Dynamic LINQ (Part 1: Using the LINQ Dynamic Query Library)

Then you can make calls like this:

```
var query = DBContext.Users.Where( "Age > 3" ).OrderBy( "Name asc" )
```

Home

PUBLIC

Stack Overflow

answered Mar 13 '13 at 15:16

Viper
1,936 1 16 36

3 Thank you Viper. This is exactly what I was looking for. To others you need to install the Nuget package `System.Linq.Dynamic` and reference it to get this to compile. – O.O. Mar 13 '13 at 15:39

1

It is not as easy as it seems. The LINQ to SQL engine parses the expression you pass to the `OrderBy` method in order to obtain the name of the property you are referencing, then uses this information to compose a plain SQL `order by` clause.

I guess that maybe it can be done by using reflection, anyway. Maybe you can get something useful from the accepted answer of this SO question.

edited May 23 '17 at 12:09

Community ♦
1 1

answered Mar 13 '13 at 15:00

Konamiman
43.1k 15 98 128

9

Try this out instead:

```
query.OrderBy(x => x.GetType().GetProperty("Name").GetValue(x, null)
```

You can't just grab the property. You need to grab the value off of that property, hence the call to `GetValue` .

answered Mar 13 '13 at 14:35

IronMan84

13.2k 15 57 76

Thank you Iron Man, but this does not work for me. If I later do a `sortedQuery.Count()` , I get an `InvalidOperationException` **Cannot order by type 'System.Object'** – O.O. Mar 13 '13 at 14:44

I would also like to mention that if I cast the result, i.e. `sortedQuery.OrderBy(x => (Product)x.GetType().GetProperty("Name").GetValue(x, null));` I get the same exception message saying it it **Cannot order by type 'MyDataContext.Product'** – O.O. Mar 13 '13 at 14:55

Of course it can't. 1) The object that `GetValue` returns is not of type Product anyway. It's only castable to the same type that the property is. 2) The LINQ has no clue how to order by `Product` anyway. You have to order based on primitive types. – IronMan84 Mar 13 '13 at 14:57

1 Because that's not sorting by the `Product` object itself in that case. It's sorting based off of *name* which is a string primitive. That's ordering that LINQ can understand. – IronMan84 Mar 13 '13 at 15:10

1 `OrderBy(x => x.Name);` is ordering by `Name` , which is a string. To know how to order products it gets more complicated. I believe you need to implement `IComparable` on your `Product` class in order to sort by OrderBy, though I'm not 100% sure. – Greg B Mar 13 '13 at 15:14