convert this LINQ expression into Lambda

Asked 10 years, 2 months ago Active 6 years, 4 months ago Viewed 30k times

14

6

Guys, I have a hard time converting this below linq expression(left join implementation) to lambda expression (for learning).

```
var result = from g in grocery
       join f in fruit on g.fruitId equals f.fruitId into tempFruit
       join v in veggie on g.vegid equals v.vegid into tempVegg
       from joinedFruit in tempFruit.DefaultIfEmpty()
       from joinedVegg in tempVegg.DefaultIfEmpty()
       select new { g.fruitId, g.vegid, fname = ((joinedFruit == null) ? string.Empty :
joinedFruit.fname), vname = ((joinedVegg == null) ? string.Empty : joinedVegg.vname) };
```

Can some one suggest me how to do this.

And i really appreciate if someone give me the excellent tutorial links for **"C# Lambdas & Linqs"**

c# linq lambda

edited Oct 6 '09 at 10:57

asked Oct 6 '09 at 10:36

RameshVel

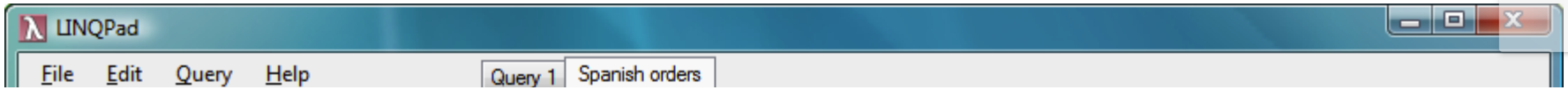
55.4k 24 155 204

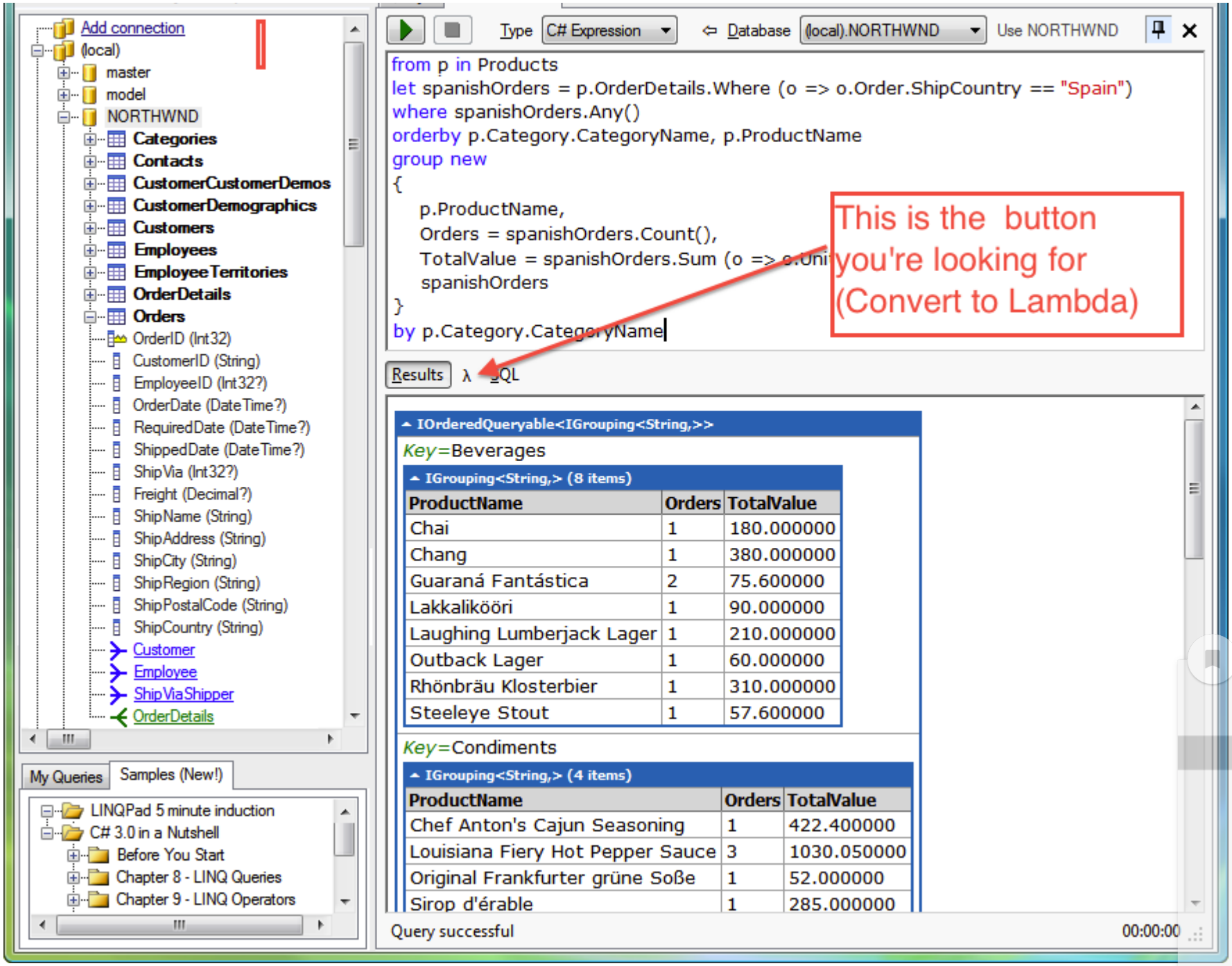
7 Answers

27

To convert a Linq query to it's Lambda equivalent:

1. Download Linqpad and run your query.
2. In the results window, click on the "λ" button in the toolbar. It's right above the Results window
3. Your query will be converted to a Lambda expression equivalent!





edited Jul 19 '13 at 15:25 answered Jul 7 '11 at 14:40

Brad Parks
44.2k 44 191 285

1 I have Linqpad but without creating a 'connection' to a service that I can query, I can not see the button for the lambda. I do not know how to just paste in a LINQ query and convert it to a lambda. Is this actually possible? – atconway Jul 19 '13 at 15:15

I just added a pic for where the button is in the UI. I don't have Linqpad on hand, but I think you need to have a runnable query there before it can be converted to a Lambda. In another post, I detailed how you can test data in Linqpad without needing a DB, here: stackoverflow.com/questions/4611031/... – Brad Parks Jul 19 '13 at 15:31

can it convert SQL to lambda or LINQ? Doesn't seem to be working – Toolkit Nov 7 '18 at 11:39

I haven't used it in a long time, but what does clicking the `SQL` button beside the `λ` do? – Brad Parks Nov 7 '18 at 11:52

12

You can take a look at 101 LINQ Samples and C# 3.0 QUERY EXPRESSION TRANSLATION CHEAT SHEET

answered Oct 6 '09 at 10:39

Dzmitry Huba
4,399 17 19

1 thanks for the links Dzmitry... the cheat sheet is useful... i already have a look at 101 samples.. – RameshVel Oct 6 '09 at 10:46

hey where would i get the source code for the 101 samples.. any idea.. i couldnt find there... – RameshVel Oct 6 '09 at 11:49

10

Here's the heuristic that I follow:

> Favor LINQ expressions over lambdas when you have joins.

I think that lambdas with joins look messy and are difficult to read.

answered Oct 6 '09 at 10:42

Jim G.
13.1k 17 84 158

thanks jim for the reply. But i wanted to do this in lambda to get familiar with.. i mentioned that already its for learning... – RameshVel Oct 6 '09 at

10:44

8

I usually use ReSharper to help me convert things to method chains and lambda's, which helps me go back and forth fairly easy.

```
    var result = from g in grocery
                 join f in fruit on g.fruitId equals f.fruitId into tempFruit
                 join v in veggie on g.vegid equals v.vegid into tempVegg
                 from joinedFruit in tempFruit.DefaultIfEmpty()
                 from joinedVegg in tempVegg.DefaultIfEmpty()
                 select new { g.fruitId, g.vegid, fname = ((joinedFruit == null) ?
string.Empty : joinedFruit.fname), vname = ((joinedVegg == null) ? string.Empty :
joinedVegg.vname) };
```

And then using ReSharper's option of convert LINQ to method chain equals the following:

```
        var result =grocery .GroupJoin(fruit, g => g.fruitId, f => f.fruitId, (g,
tempFruit) => new {g, tempFruit})
                            .GroupJoin(veggie, @t => @t.g.vegid, v => v.vegid, (@t,
tempVegg) => new {@t, tempVegg})
                            .SelectMany(@t => @t.@t.tempFruit.DefaultIfEmpty(), (@t,
joinedFruit) => new {@t, joinedFruit})
                            .SelectMany(@t => @t.@t.tempVegg.DefaultIfEmpty(),(@t,
joinedVegg) =>
                                new
                                    {
                                        @t.@t.@t.g.fruitId,
                                        @t.@t.@t.g.vegid,
                                        fname = ((@t.joinedFruit == null) ? string.Empty
: @t.joinedFruit.fname),
                                        vname = ((joinedVegg == null) ? string.Empty :
joinedVegg.vname)
                                    });
```

Granted the output is less then desirable, but It at least helps in starting somewhere on understanding the syntax.

answered Oct 6 '09 at 12:39

Mark Coleman
37.6k 8 77 102

thanks Mark for your response.. i will run it over here and will let you know if that works.. – RameshVel Oct 6 '09 at 13:24

4

Here's how you might write this query in lambda:

```
var customers = new List {
new Customer { CompanyId = "AC", CustomerId = "Customer1" },
new Customer { CompanyId = "not-AC", CustomerId = "Customer2" },
};

var userCustomers = new List {
new UserCustomer { CompanyId = "AC", CustomerId = "Customer1", User = "not-admin"
},
new UserCustomer { CompanyId = "AC", CustomerId = "Customer1", User = "admin" },
new UserCustomer { CompanyId = "AC", CustomerId = "Customer2", User = "not-admin"
},
new UserCustomer { CompanyId = "AC", CustomerId = "Customer2", User = "admin" },
new UserCustomer { CompanyId = "not-AC", CustomerId = "Customer1", User = "not-
admin"     },
new UserCustomer { CompanyId = "not-AC", CustomerId = "Customer1", User = "admin"
},
new UserCustomer { CompanyId = "not-AC", CustomerId = "Customer2", User = "not-
admin" },
new UserCustomer { CompanyId = "not-AC", CustomerId = "Customer2", User = "admin" }
};
```

Using query expression

```
var query =
from c in customers
join uc in userCustomers on
new { c.CompanyId, c.CustomerId } equals new { uc.CompanyId, uc.CustomerId }
where c.CompanyId == "AC" && uc.User == "admin"
select c;
```

Using lambda expressions

```
var lambda =  customers.Where(c => c.CompanyId == "AC") // inner sequence
.Join(userCustomers.Where(uc => uc.User == "admin"), // outer sequence
c => new { c.CompanyId, c.CustomerId }, // inner key selector
uc => new { uc.CompanyId, uc.CustomerId }, // outer key selector
(c, uc) => c);
```

Both approach yields the same result (customer with company Id “AC” and customer Id “Customer1”), but as you can see, lambda expression is much harder to write and read!

Hope this helps!

edited Apr 13 '12 at 5:27

menjaraz
7,339 4 34 76

answered Apr 13 '12 at 5:04

Sunandan Dutt
47 3

3

Download LINQPad; it comes with built-in samples for learning LINQ.

answered Oct 6 '09 at 10:40

Mitch Wheat
266k 37 421 507

i thought that was only the paid version that had loads of samples etc – Andrew Oct 6 '09 at 10:53

The free version comes with samples too. – Mitch Wheat Oct 6 '09 at 11:18

3

Use Reflector .NET :)

answered Oct 6 '09 at 10:41

leppie
103k 16 178 287