Am I missing something about LINQ?

Asked 16 years, 4 months ago Modified 14 years, 6 months ago

Viewed 2k times

20

I seem to be missing something about LINQ. To me, it looks like it's taking some of the elements of SQL that I like the least and moving them into the C# language and using them for other things.

I mean, I could see the benefit of using SQL-like statements on things other than databases. But if I wanted to write SQL, well, why not just write SQL and keep it out of C#? What am I missing here?

c# sql linq

Share

Improve this question

Follow

edited Aug 21, 2008 at 21:59

palmsey

5,832 ● 3 ● 38 ● 41

asked Aug 21, 2008 at 21:54

Jason Baker

198k ● 138 ● 382 ● 520

6 Answers

Sorted by: Highest score (default)

40

LINQ is not about SQL. LINQ is about being apply functional programming paradigmns on objects.

LINQ to SQL is an ORM built ontop of the LINQ foundation, but LINQ is much more. I don't use LINQ to SQL, yet I use LINQ all the time.

Take the task of finding the intersection of two lists:

Before LINQ, this tasks requires writing a nested foreach that iterates the small list once for every item in the big list O(N*M), and takes about 10 lines of code.

```
foreach (int number in list1)
{
    foreach (int number2 in list2)
    {
        if (number2 == number)
        {
            returnList.add(number2);
        }
    }
}
```

Using LINQ, it does the same thing in one line of code:

```
var results = list1.Intersect(list2);
```

You'll notice that doesn't look like LINQ, yet it is. You don't need to use the expression syntax if you don't want to.

Share Improve this answer Follow

answered Aug 21, 2008 at 21:59

FlySwat
175k ● 75 ● 248 ● 314

This was good thank you. I'm interested, in your two code examples would the LINQ version be any faster the the "original" version? – Frank V Nov 14, 2008 at 18:10

LINQ isn't going to magically take the time complexity out of that operation though. So, yes it looks nicer but if it is optimized, it is only a little. – mmcdole Feb 9, 2009 at 0:48

1 Which is nice, because I hate the expression syntax. – Instance Hunter Feb 28, 2009 at 5:57

That is perhaps the weirdest thing i've seen. "Oh, let's use LINQ, but instead of making my code most readable to future developers by using the builtin to c# syntax, let's go ahead and make a long chain of function calls! great!" – RCIX Dec 5, 2009 at 9:13

@Derrick Turk: True enough, but then you can't using things like `yield return` or iterator blocks :) – RCIX Apr 22, 2010 at 1:29

20

Before:

```
// Init Movie
m_ImageArray = new Image[K_NB_IMAGE];

Stream l_ImageStream = null;
Bitmap l_Bitmap = null;

// get a reference to the current assembly
Assembly l_Assembly = Assembly.GetExecutingAssembly();

// get a list of resource names from the manifest
string[] l_ResourceName = l_Assembly.GetManifestResour
```

```
foreach (string l_Str in l_ResourceName)
{
    if (l_Str.EndsWith(".png"))
    {
        // attach to stream to the resource in the man
        l_ImageStream = l_Assembly.GetManifestResource
        if (!(null == l_ImageStream))
        {
            // create a new bitmap from this stream an
            // add it to the arraylist
            l_Bitmap = Bitmap.FromStream(l_ImageStream
            if (!(null == l_Bitmap))
            {
                int l_Index = Convert.ToInt32(l_Str.Su
2));
                l_Index -= 1;
                if (l_Index < 0) l_Index = 0;
                if (l_Index > K_NB_IMAGE) l_Index = K_
                m_ImageArray[l_Index] = l_Bitmap;
            }
            l_Bitmap = null;
            l_ImageStream.Close();
            l_ImageStream = null;
        } // if
    } // if
} // foreach
```

After:

```
Assembly l_Assembly = Assembly.GetExecutingAssembly();

//Linq is the tops
m_ImageList = l_Assembly.GetManifestResourceNames()
    .Where(a => a.EndsWith(".png"))
    .OrderBy(b => b)
    .Select(c => l_Assembly.GetManifestResourceStream(
    .Where(d => d != null)  //ImageStream not null
    .Select(e => Bitmap.FromStream(e))
    .Where(f => f != null)  //Bitmap not null
    .ToList();
```

Or, alternatively (*query syntax*):

```
Assembly l_Assembly = Assembly.GetExecutingAssembly();

//Linq is the tops
m_ImageList = (
    from resource in l_Assembly.GetManifestResourceNam
    where resource.EndsWith(".png")
    orderby resource
    let imageStream = l_Assembly.GetManifestResourceSt
    where imageStream != null
    let bitmap = Bitmap.FromStream(imageStream)
    where bitmap != null)
    .ToList();
```

Share Improve this answer edited Jun 7, 2010 at 4:46

Follow

answered Sep 17, 2008 at 7:10

Benjol

66.4k ● 55 ● 192 ● 275

3 The before sample is more verbose but it has the advantage of being debug-able. It can be difficult to analyse and fix complex linq statements because they hide so much from the developer. – Phil Gan Jun 7, 2010 at 15:24

2 @Phil, yes and no, the more lines of code you have, the more there is to debug too! Having said that, that is why I prefer the method syntax (first of the After examples), because it is easy to splice the method chain into two at any point. – Benjol Jun 8, 2010 at 4:41

@Benjol: I'd never thought of doing that... So you can break down a long linq statement to see what each method returns or combine several short linq statements into one when

you've verified they do as you expect. That's interesting! – Phil Gan Jun 8, 2010 at 8:12

In the Before version, you explicitly Close each image stream. Does this happen underneath the hood in the LINQ version? – I. J. Kennedy Jun 24, 2010 at 19:35

@I. J, good question. I think it's a bug in my code. Linq does lots of magic stuff, but I don't think it goes that far. This is probably worth another question actually... (maybe here?: stackoverflow.com/questions/1751153/…) – Benjol Jun 25, 2010 at 5:32

5

So the really, really big deal about LINQ has nothing to do with Linq to SQL. It's about the enhancements it brought to the C# language itself.

Share Improve this answer

Follow

answered Aug 21, 2008 at 22:18

ScottKoon

3,493 ● 6 ● 28 ● 29

5

LINQ is not just an ORM system, as Jonathan pointed out it brings a lot of functional programming elements to C#. And it lets you do a lot of "database-y" things in regular C# code. It's difficult to explain just how incredibly powerful that can be. Consider how much having solid, well designed generic data structures (such as list, stack, dictionary/hash, etc.) included in common frameworks has improved the state of development in modern languages. Precisely because using these data structures

is very common and reducing the intellectual overhead of using them is a huge benefit. LINQ doesn't do anything you can't do yourself, but it makes a lot of operations a lot more straightforward and a lot easier.

Consider the time-honored example of removing duplicates from a non-ordered list. In a lower level language like C or C++ you'd probably have to sort the list and maintain two indices into the list as you removed dupes. In a language with hashes (Java, C#, Javascript, Perl, etc.) you could create a hash where the keys are the unique values, then extract the keys into a new list. With LINQ you could just do this:

```
int[] data = { 0, 1, 3, 3, 7, 8, 0, 9, 2, 1 };

var uniqueData = data.GroupBy(i => i).Select(g => g.Ke
```

Share Improve this answer

Follow

answered Aug 21, 2008 at 23:20

Wedge

19.8k ● 7 ● 50 ● 71

3 Or even better: data.Distinct(); – Nathan Baulch Dec 1, 2008 at 17:42

4

Because linq is really monads in sql clothing, I'm using it on a project to make asynchronous web requests with the continuation monad, and it's proving to work really well!

Check out these articles:
http://www.aboutcode.net/2008/01/14/Async+WebRequest+Using+LINQ+Syntax.aspx
http://blogs.msdn.com/wesdyer/archive/2008/01/11/the-marvels-of-monads.aspx

From the first article:

```
var requests = new[]
{
    WebRequest.Create("http://www.google.com/"),
    WebRequest.Create("http://www.yahoo.com/"),
    WebRequest.Create("http://channel9.msdn.com/")
};

var pages = from request in requests
            select
                from response in request.GetRespon
                let stream = response.GetResponseS
                from html in stream.ReadToEndAsync
                select new { html, response };

foreach (var page in pages)
{
    page(d =>
    {
        Console.WriteLine(d.response.ResponseUri.T
        Console.WriteLine(d.html.Substring(0, 40))
        Console.WriteLine();
    });
}
```

Share Improve this answer

Follow

answered Mar 13, 2009 at 3:02

Jacob Stanley

4,814 ● 3 ● 34 ● 37

2

The point is that LINQ integrates your queries into your primary programming language, allowing your IDE to provide you with some facilities (Intellisense and debug support, for example) that you otherwise would not have, and to allow the compiler to type-check your SQL code (which is impossible with a normal string query).

Share Improve this answer

Follow

answered Aug 21, 2008 at 21:56

TheSmurf

15.6k ● 3 ● 41 ● 48

But still why would you want to write sql code in LINQ? How would you performance tune it? As a database person, I'm with the questioner. I just don't get why you would want to do it from a database perspective. The queries I've seen from LINQ look bloated to me. – HLGEM Dec 5, 2008 at 20:23

1 If you're using an ORM (*any* ORM) the queries are going to be bloated. If you want bare-metal, you should be using SQL through ADO.NET (which is always a possibility in some critical section of code , even if you use LINQ primarily).
– TheSmurf Dec 10, 2008 at 18:50

Another reason to use linq is that, however bloated the query may be, the SQL engine authors might have a smaller set of query patterns to tune internally which would be nice.
– Michael Haren Feb 17, 2009 at 20:52