# Delegates and Lambdas and LINQ, Oh My!

Asked 15 years, 11 months ago Modified 15 years, 11 months ago

Viewed 4k times

12

As a fairly junior developer, I'm running into a problem that highlights my lack of experience and the holes in my knowledge. Please excuse me if the preamble here is too long.

I find myself on a project that involves my needing to learn a number of new (to me) technologies, including LINQ (to OBJECTS and to XML for purposes of this project) among others. Everything I've read to this point suggests that to utilize LINQ I'll need to fully understand the following (Delegates, Anonymous Methods and Lambda Expressions).

OK, so now comes the fun. I've CONSUMED delegates in the past as I have worked with the .NET event model, but the majority of the details have been hidden from me (thanks Microsoft!). I understand that on a basic level, delegate instances are pointers to methods (a gross over-simplification, I know).

I understand that an anonymous method is essentially an in-line unnamed method generally (if not exclusively) created as a target for a delegate.

I also understand that lambdas are used in varying ways to simplfy syntax and can be used to point a simple anonymous method to a delegate.

Pardon me if my any of my descriptions are WAY off here, this is the basic level to which I understand these topics.

So, the challenge:

1. Can anyone tell me if at least on a basic level if my understanding of these items is even close? I'm not looking for complex esoteric minutiae, just the basics (for now).
2. To what degree do I need to truly understand these concepts before applying LINQ in a project to reasonable effect? I want to understand it fully and am willing to spend the time. I simply may not HAVE the time to fully grok all of this stuff before I need to produce some work.
3. Can anyone point me to some good articles that explain these subjects and apply them to "real world" examples so that I can get my head around the basics of the topics and application of them? What I mean by real world, is how might I use this in the context of "Customers and Invoices" rather than abstract "Vectors and Shapes" or "Animals and Cows". The scenario can be somewhat contrived for demonstration purposes, but hopefully not strictly academic. I have found a number of examples on-line and in books, but few seem to be "Plain English" explanations.

Thank you all in advance for your patience, time and expertise.

c# linq delegates lambda anonymous-methods

Share

Improve this question

Follow

edited Jan 13, 2009 at 16:11

Jon Skeet

**1.5m** ● 889 ● 9.3k ● 9.3k

asked Jan 13, 2009 at 16:01

Steve Brouillard

**3,266** ● 5 ● 43 ● 60

## 3 Answers

Sorted by: Highest score (default)

6

[Where can i find a good in depth guide to C# 3?]

1) Your knowledge so far seems ok. Lambda expressions are turned into anonymous methods or System.Linq.Expressions.Expression's, depending on context. Since you aren't using a database technology, you don't need to understand expressions (all lambdas will be anonymous methods). You didn't list Extension methods, but those are very important (and easy) to understand. Make sure you see how to apply an extension method to an interface - as all the functionality in linq comes from System.Linq.Enumerable - a collection of extention methods against IEnumerable(Of T).

2) You don't need a deep understanding of lambdas.

The arrow syntax ( => ) was the biggest hurdle for me. The arrow separates the signature and the body of the lambda expression.

Always remember : **Linq methods are not executed until enumerated.**

Watch out for using loop variables in a lambda. This is a side effect from deferred execution that is particularly tricky to track down.

3) Sure, Here are some of my answers that show linq method calls - some with xml.

- List splitting
- Simple Xml existence search
- Xml projection - shape change

Share Improve this answer

Follow

edited May 23, 2017 at 12:30

Community Bot

1 ● 1

answered Jan 13, 2009 at 16:06

Amy B

110k ● 21 ● 139 ● 190

I would clarify your bit in bold a bit. The method itself is executed - but if a sequence is returned, *that* is lazily evaluated. In particular, things like Max, Min Count etc compute the answer immediately! Personally I find that

imaginng the implementation of Select() etc is helpful.
– Jon Skeet Jan 13, 2009 at 16:30

Thanks very much for the links. Together with Jon's answer I think I will be well on my way here. I've got a lot to wrangle in the next couple of days, but I think I can manage it.
– Steve Brouillard Jan 13, 2009 at 17:27

5

1) Those descriptions sound pretty accurate to me. Sometimes anonymous methods and lambda expressions will need to create a new type to put the target of the delegate in, so they can act as closures.

2/3) I would read up a bit until you're happy with delegates, anonymous methods and lambda expressions. I dedicate a chapter to the delegate-related changes in each of C# 2.0 and C# 3.0 in C# in Depth, although of course other books go into detail too. I have an article as well, if that helps.

As for examples - delegates are used for many different purposes. They're all different ways of looking at the same functionality, but they can *feel* very different:

- Providing the code to call when you start a new thread
- Reacting to UI events
- Providing the filter, selection, ordering etc for a LINQ query
- Providing a callback for when an asynchronous operation has finished

If you have any specific situations you'd like an example of, that would be easier to answer.

EDIT: I should point out that it's good news that you're only working with LINQ to Objects and LINQ to XML at the moment, as that means you don't need to understand expression trees yet. (They're cool, but one step at a time...) LINQ to XML is really just an XML API which works nicely with LINQ - from what I remember, the only times you'll use delegates with LINQ to XML are when you're actually calling into LINQ to Objects. (That's very nice to do, admittedly - but it means you can reuse what you've already learned.)

As you've already got C# in Depth, chapters 10 and 11 provide quite a few examples of *using* lambda expressions (and query expressions which are translated into lambda expressions) in LINQ. Chapter 5 has a few different examples of delegate use.



edited Jan 13, 2009 at 16:16

answered Jan 13, 2009 at 16:09

Jon Skeet
**1.5m** ● 889 ● 9.3k ● 9.3k

Well, the good news is that I HAVE your book. As it turns out, I have the Dead Tree Edition, the free eBook that comes with buying that AND access to it via a recently purchased Safari

subscription. Now, let's see if my mind is nimble enough to UNDERSTAND! – Steve Brouillard Jan 13, 2009 at 16:12

That's good news on all fronts :) Please feel very free to ask as many questions about specific bits as you want, whether here, by private email (skeet@pobox.com) or on the Manning forum. Specific questions are always easier to answer :) – Jon Skeet Jan 13, 2009 at 16:13

Thanks very much for the offer. I may be taking you up on it soon. I'm going to get started on your book right away and read it together with LINQ in Action. – Steve Brouillard Jan 13, 2009 at 17:27

1

Read this...

http://linqinaction.net/

..and all you're question will be answered!!!

Share Improve this answer

Follow

answered Jan 13, 2009 at 16:10

matt_dev

**5,246** ● 5 ● 35 ● 44