Using LINQ to remove elements from a List<T>

604

Say that I have LINQ query such as:

```
var authors = from x in authorsList
              where x.firstname == "Bob"
              select x;
```

103

Given that `authorsList` is of type `List<Author>`, how can I delete the `Author` elements from `authorsList` that are returned by the query into `authors`?

Or, put another way, how can I delete all of the firstname's equalling Bob from `authorsList`?

Note: This is a simplified example for the purposes of the question.

c# .net linq list

edited Jan 19 '16 at 19:50
John M
6,458 27 79 128

asked May 12 '09 at 15:56
TK.
19.7k 45 108 141

15 Answers

1040

Well, it would be easier to exclude them in the first place:

```
authorsList = authorsList.Where(x => x.FirstName != "Bob").ToList();
```

However, that would just change the value of `authorsList` instead of removing the authors from the previous collection. Alternatively, you can use `RemoveAll`:

If you really need to do it based on another collection, I'd use a HashSet, RemoveAll and Contains:

```
var setToRemove = new HashSet<Author>(authors);
authorsList.RemoveAll(x => setToRemove.Contains(x));
```

answered May 12 '09 at 16:01
Jon Skeet
1114k 708 8116 8532

13 What's the reason for using HashSet for another collection? – 123 456 789 0 Aug 7 '12 at 1:51

51 @LeoLuis: It makes the `Contains` check fast, and ensures you only evaluate the sequence once. – Jon Skeet Aug 7 '12 at 1:57

2 @LeoLuis: Yes, building a HashSet from a sequence only evaluates it once. Not sure what you mean by "weak collection set". – Jon Skeet Aug 7 '12 at 3:21

2 @AndréChristofferAndersen: What do you mean by "outdated"? It still works. If you've got a `List<T>` , it's fine to use it. – Jon Skeet Apr 28 '13 at 12:08

4 @AndréChristofferAndersen: It would be better to use `authorsList = authorsList.Where(x => x.FirstName != "Bob")` – Jon Skeet May 26 '13 at 11:30

122

It'd be better to use List<T>.RemoveAll to accomplish this.

```
authorsList.RemoveAll((x) => x.firstname == "Bob");
```

answered May 12 '09 at 16:03
Reed Copsey
477k 60 1001 1286

6 @Reed Copsey: The lambda parameter in your example is enclosed in parentheses, i.e., (x). Is there a technical reason for this? Is it considered good practice? – Matt Davis Sep 10 '09 at 5:56

43

If you really need to remove items then what about Except()?
You can remove based on a new list, or remove on-the-fly by nesting the Linq.

```
var authorsList = new List<Author>()
{
    new Author{ Firstname = "Bob", Lastname = "Smith" },
    new Author{ Firstname = "Fred", Lastname = "Jones" },
    new Author{ Firstname = "Brian", Lastname = "Brains" },
    new Author{ Firstname = "Billy", Lastname = "TheKid" }
};

var authors = authorsList.Where(a => a.Firstname == "Bob");
authorsList = authorsList.Except(authors).ToList();
authorsList = authorsList.Except(authorsList.Where(a=>a.Firstname=="Billy")).ToList();
```

edited Apr 12 '12 at 16:01
abatishchev
71.3k 70 269 399

answered Nov 19 '10 at 13:54
BlueChippy
2,432 12 63 116

`Except()` is the only way to go in middle of LINQ-statement. `IEnumerable` doesn't have `Remove()` nor `RemoveAll()` . – Jari Turkia Apr 11 at 6:43

26

You cannot do this with standard LINQ operators because LINQ provides query, not update support.

But you can generate a new list and replace the old one.

```
var authorsList = GetAuthorList();

authorsList = authorsList.Where(a => a.FirstName != "Bob").ToList();
```

Or you could remove all items in `authors` in a second pass.

```
var authorsList = GetAuthorList();

var authors = authorsList.Where(a => a.FirstName == "Bob").ToList();
```

```
    authorList.Remove(author);
}
```

answered May 12 '09 at 16:02
Daniel Brückner
51.6k 10 86 133

Incorrect. `RemoveAll()` **does** update the list in-place. – Shai Cohen Apr 4 '14 at 17:41

9 `RemoveAll()` is not a LINQ operator. – Daniel Brückner Apr 4 '14 at 17:50

My apologies. You are 100% correct. Unfortunately, I can't seem to reverse my downvote. Sorry about that. – Shai Cohen Apr 4 '14 at 21:16

2 No problem, I dont't care about two points more or less. – Daniel Brückner Apr 4 '14 at 21:18

`Remove` is also a `List < T >` method, not a System.Linq.Enumerable method. – DavidRR Jul 29 '14 at 1:42

20

Simple solution:

```
static void Main()
{
    List<string> myList = new List<string> { "Jason", "Bob", "Frank", "Bob" };
    myList.RemoveAll(x => x == "Bob");

    foreach (string s in myList)
    {
        //
    }
}
```

edited Apr 3 '11 at 18:40
abatishchev
71.3k 70 269 399

answered May 12 '09 at 16:06
CodeLikeBeaker
13.7k 12 61 91

how to remove "Bob" and "Jason" I mean multiple in string list? – Neo Mar 25 '18 at 8:20

15

performance check :)

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Text;

namespace ListRemoveTest
{
    class Program
    {
        private static Random random = new Random( (int)DateTime.Now.Ticks );

        static void Main( string[] args )
        {
            Console.WriteLine( "Be patient, generating data..." );

            List<string> list = new List<string>();
            List<string> toRemove = new List<string>();
            for( int x=0; x < 1000000; x++ )
            {
                string randString = RandomString( random.Next( 100 ) );
                list.Add( randString );
                if( random.Next( 1000 ) == 0 )
                    toRemove.Insert( 0, randString );
            }

            List<string> l1 = new List<string>( list );
            List<string> l2 = new List<string>( list );
            List<string> l3 = new List<string>( list );
            List<string> l4 = new List<string>( list );

            Console.WriteLine( "Be patient, testing..." );

            Stopwatch sw1 = Stopwatch.StartNew();
            l1.RemoveAll( toRemove.Contains );
            sw1.Stop();

            Stopwatch sw2 = Stopwatch.StartNew();
            l2.RemoveAll( new HashSet<string>( toRemove ).Contains );
            sw2.Stop();

            Stopwatch sw3 = Stopwatch.StartNew();
            l3 = l3.Except( toRemove ).ToList();
            l4 = l4.Except( new HashSet<string>( toRemove ) ).ToList();
            sw3.Stop();


            Console.WriteLine( "L1.Len = {0}, Time taken: {1}ms", l1.Count,
sw1.Elapsed.TotalMilliseconds );
            Console.WriteLine( "L2.Len = {0}, Time taken: {1}ms", l1.Count,
sw2.Elapsed.TotalMilliseconds );
            Console.WriteLine( "L3.Len = {0}, Time taken: {1}ms", l1.Count,
sw3.Elapsed.TotalMilliseconds );
            Console.WriteLine( "L4.Len = {0}, Time taken: {1}ms", l1.Count,
sw3.Elapsed.TotalMilliseconds );

            Console.ReadKey();
        }


        private static string RandomString( int size )
        {
            StringBuilder builder = new StringBuilder();
            char ch;
            for( int i = 0; i < size; i++ )
            {
                ch = Convert.ToChar( Convert.ToInt32( Math.Floor( 26 *
random.NextDouble() + 65 ) ) );
                builder.Append( ch );
            }

            return builder.ToString();
        }
    }
}
```

Results below:

```
Be patient, generating data...
Be patient, testing...
L1.Len = 985263, Time taken: 13411.8648ms
L2.Len = 985263, Time taken: 76.4042ms
L3.Len = 985263, Time taken: 340.6933ms
L4.Len = 985263, Time taken: 340.6933ms
```

As we can see, best option in that case is to use ***RemoveAll(HashSet)***

151 1 3

This code: "l2.RemoveAll(new HashSet<string>(toRemove).Contains);" should not compile... and if your tests are correct then they just second what Jon Skeet already suggested. – Pascal Jul 24 '14 at 18:37

1 `l2.RemoveAll( new HashSet<string>( toRemove ).Contains );` compiles fine just FYI – AzNjoE Mar 1 '16 at 17:44

8

This is a very old question, but I found a really simple way to do this:

```
authorsList = authorsList.Except(authors).ToList();
```

Note that since the return variable `authorsList` is a `List<T>` , the `IEnumerable<T>` returned by `Except()` must be converted to a `List<T>` .

edited Jul 29 '14 at 2:53
DavidRR
10k 11 64 139

answered Apr 26 '13 at 9:52
Carlos Martinez T
5,977 1 28 38

7

You can remove in two ways

```
var output = from x in authorsList
             where x.firstname != "Bob"
             select x;
```

or

```
var authors = from x in authorsList
              where x.firstname == "Bob"
              select x;

var output = from x in authorsList
             where !authors.Contains(x)
             select x;
```

edited Apr 12 '12 at 16:02
abatishchev
71.3k 70 269 399

answered Sep 7 '10 at 12:02
AsifQadri
2,048 1 16 27

How can I check for "Bob" or "Billy"? – Si8 Jan 3 '17 at 15:44

6

Say that `authorsToRemove` is an `IEnumerable<T>` that contains the elements you want to remove from `authorsList` .

Then here is another very simple way to accomplish the removal task asked by the OP:

```
authorsList.RemoveAll(authorsToRemove.Contains);
```

edited Jul 29 '14 at 2:38
DavidRR
10k 11 64 139

answered Sep 30 '13 at 21:15
atconway
12k 19 120 197

5

I think you could do something like this

```
    authorsList = (from a in authorsList
                   where !authors.Contains(a)
                   select a).ToList();
```

Although I think the solutions already given solve the problem in a more readable way.

answered May 12 '09 at 16:41
ebrown
526 1 6 13

Below is the example to remove the element from the list.

```
returns boolean value
 var result1 = items.RemoveAll(lst => lst == 3);// Remove all the matched elements and
returns count of removed element
 items.RemoveAt(3);//Removes the elements at the specified index
```

answered Aug 25 '16 at 18:19
Sheo Dayal Singh
693 6 7

0

LINQ has its origins in functional programming, which emphasises immutability of objects, so it doesn't provide a built-in way to update the original list in-place.

Note on immutability (taken from another SO answer):

Here is the definition of immutability from Wikipedia (link)

"In object-oriented and functional programming, an immutable object is an object whose state cannot be modified after it is created."

edited Jan 19 '16 at 19:53
John M
6,458 27 79 128

answered May 12 '09 at 16:03
Samuel Jack
26k 12 104 147

0

i think you just have to assign the items from Author list to a new list to take that effect.

```
//assume oldAuthor is the old list
Author newAuthorList = (select x from oldAuthor where x.firstname!="Bob" select
x).ToList();
oldAuthor = newAuthorList;
newAuthorList = null;
```

answered Jan 28 '16 at 5:36
aj go
91 2 11

0

```
authorsList = authorsList.Where(x => x.FirstName != "Bob").<do_some_further_Linq>;
```

or

```
authorsList = authorsList.Where(x => !setToRemove.Contains(x)).<do_some_further_Linq>;
```

answered Jun 15 '18 at 13:48
Zbigniew Wiadro
957 15 21

-2

Is very simple:

```
authorsList.RemoveAll((x) => x.firstname == "Bob");
```

answered Feb 24 '16 at 12:06
Sandro Z.
28 7

12 Why did you write this answer 6 years after the accepted answer, which already contains what you are saying? – EluciusFTW Nov 15 '16 at 8:28