check if record is last or first in list with linq

3

I have a **list of objects**. I want to determine when the user will get the **first or last object in the list**, that way I can disable some buttons on the page.

For example I might have some boolean and if the object requested is last or first in the list, then it will return `true`, otherwise `false`.

2

Any idea?

c# asp.net linq

edited Feb 4 '13 at 12:58

mipe34
4,927 3 21 35

asked Feb 4 '13 at 12:48

Laziale
3,086 35 107 199

Some context is missing which may affect the answer. How are you currently returning the requested object? What does your code look like? Are you already searching the list for the object? If so, how? – Matthew Watson Feb 4 '13 at 12:58

3 Answers

4

```
public static class IEnumerableExtensions
{
    public static bool IsLast<T>(this IEnumerable<T> items
    {
        var last =  items.LastOrDefault();
        if (last == null)
            return false;
        return item.Equals(last); // OR Object.ReferenceEq
    }

    public static bool IsFirst<T>(this IEnumerable<T> item
    {
        var first =  items.FirstOrDefault();
        if (first == null)
            return false;
        return item.Equals(first);
    }

    public static bool IsFirstOrLast<T>(this IEnumerable<T
    {
        return items.IsFirst(item) || items.IsLast(item);
    }
}
```

You can use it like

```
IEnumerable<User> users = // something
User user = // something

bool isFirstOrLast = users.IsFirstOrLast(user);
```

edited Feb 4 '13 at 12:59

answered Feb 4 '13 at 12:53

Mehmet Ataş

7,310 5 37 64

3 Used indiscriminately, this might have disastrous performance

3 Might also lead to false positives if the first/last item is not distinct in the sequence – spender Feb 4 '13 at 13:09

First/Last extensions check if the `IEnumerable` instance is `IList`. If so, they use `list[0]` or `list[list.Count - 1]`. But if it is not `IList` yes, there will be some overhead. On the other hand, if you need to access some certain elements of a collection and you choose a collection implementation which does not support access by index, it would be a general design flaw. – Mehmet Ataş Feb 4 '13 at 19:30

8

If your list of objects is indeed a `List`, it might be better to use it explicitly (adapted the Mehmet Ataş's answer):

```
static class ListExtensions
{
  public static bool IsFirst<T>(this List<T> items, T item
  {
    if (items.Count == 0)
      return false;
    T first =  items[0];
    return item.Equals(first);
  }

  public static bool IsLast<T>(this List<T> items, T item)
  {
    if (items.Count == 0)
      return false;
    T last =  items[items.Count-1];
    return item.Equals(last);
  }
}
```

This way you eliminate the LINQ overhead (it's not much, but it's significant). However, your code must use `List<T>` for this to work.

edited Jun 14 '13 at 14:19

answered Feb 4 '13 at 13:02

SWeko
25.8k 6 56 94

2 IList<T> interface would work as well. – liang Feb 26 '16 at 9:08

adapted the Mehmet Ataş's answer - this is so unnecessary. – t3chb0t Aug 21 '16 at 17:42

@t3chb0t, just giving credit where it's due... – SWeko Aug 22 '16 at 7:11

I mean the implementation not the credit. `First` / `Last` already can `IList` so implementing it with a `List<>` is rather redudant. – t3chb0t Aug 22 '16 at 7:13

@t3chb0t, Those work as `IEnumerable<T>` extension methods, these will be marginally faster, as they work with the `List` / `IList` directly. – SWeko Aug 24 '16 at 11:59

1

```
var yourObject = yourList[0];
if(list.Count > 0)
    if(yourObject == list.First() || yourobject == list.La
    {
     //item is either first or last
    }
```

But remember to check if the list contains atleast 1 item, otherwise you will end up with exception from First, Last.

The above will compare the reference for the objects, you can compare their values by implementing a custom IComparable or you may compare their values.

answered Feb 4 '13 at 12:52

Habib
184k 23 321 362

2 ...also worth being aware that in most/all cases, `.Last()` causes a full enumeration of the source sequence. – spender Feb 4 '13 at 12:52

2 If the `IEnumerable` is indeed a `List`, `Last()` will just return `list[list.Count-1]` – SWeko Feb 4 '13 at 12:57

1 @Downvoter, care to comment ? – Habib Feb 4 '13 at 13:03