How to remove an item from a list in Scala having only its index?

Asked 7 years, 10 months ago Active 1 year, 11 months ago Viewed 29k times

23

4

I have a list as follows:

```
val internalIdList: List[Int] = List()

internalIdList = List(11, 12, 13, 14, 15)
```

From this list would remove the third element in order to obtain:

```
internalIdList = List(11, 12, 14, 15)
```

I can not use a `ListBuffer`, are obliged to maintain the existing structure. How can I do?

Thanks to all

list scala filter scala-2.10

Share Improve this question Follow

edited Aug 23 '16 at 18:45
elm
18.7k 11 59 106

asked Sep 17 '13 at 10:21
YoBre
2,390 5 22 37

7 Answers

Active Oldest Votes

20

Simply use

```
val trunced = internalIdList.take(index) ++ internalIdList.drop(index + 1)
```

This will also work if index is larger than the size of the list (It will return the same list).

Share Improve this answer Follow

edited Sep 1 '19 at 10:51
wvdz
15.4k 4 46 82

answered Sep 17 '13 at 10:45
Shadowlands
14.6k 4 41 41

2 The if/else is useless: take and drop will behave as expected if the given index is greater than the size. – Nicolas Sep 18 '13 at 10:02

There is a `.patch` method on `Seq`, so in order to remove the third element you could simply do this:

51

```
List(11, 12, 13, 14, 15).patch(2, Nil, 1)
```

Which says: Starting at index **2**, please remove **1** element, and replace it with **Nil**.

Knowing this method in depth enables you to do so much more than that. You can swap out any sublist of a list with arbitrary other.

Share Improve this answer Follow

edited Jul 23 '18 at 22:45

answered Jul 21 '14 at 10:50

Rok Kralj
41.4k 10 63 78

13

An idiomatic way to do it is to zip the value with their index, filter, and then project the value again:

```
scala> List(11,12,13,14,15).zipWithIndex.filter(_._2 != 2).map(_._1)
res0: List[Int] = List(11, 12, 14, 15)
```

But you can also use `splitAt` :

```
scala> val (x,y) = List(11,12,13,14,15).splitAt(2)
x: List[Int] = List(11, 12)
y: List[Int] = List(13, 14, 15)

scala> x ++ y.tail
res5: List[Int] = List(11, 12, 14, 15)
```

Share Improve this answer Follow

answered Sep 17 '13 at 11:42

Nicolas
23.5k 4 57 64

4

If you insist on using the oldschool method, use collect:

```
List(1,2,3,4).zipWithIndex.collect { case (a, i) if i != 2 => a }
```

However, I still prefer the method in my other answer.

Share Improve this answer Follow

answered Mar 23 '16 at 16:43

Rok Kralj
41.4k 10 63 78

1 This is amazing! – Kevin eyeson Sep 24 '20 at 22:22

Glad you like it, still, I suggest my .patch() answer. ;) – Rok Kralj Sep 29 '20 at 22:07

```
(internalIdList.indices.collect { case i if i != 3 => internalList(i) }).toList
```

1

To generalise this...

```
def removeIndex[A](s: Seq[A], n: Int): Seq[A] = s.indices.collect { case i if i != n =>
s(i) }
```

Although this will often return a Vector, so you would need to do

```
val otherList = removeIndex(internalIdList, 3).toList
```

If you really wanted a list back.

Shadowlands has a solution which *tends* to be faster for linear sequences. This one will be faster with indexed sequences.

Share Improve this answer Follow edited Sep 17 '13 at 14:43 answered Sep 17 '13 at 14:37

itsbruce
4,671 24 34

1

A generic function that implements Nicolas' first solution:

```
def dropIndex[T](list: List[T], idx: Int): List[T] =
  list.zipWithIndex.filter(_._2 != idx).map(_._1)
```

Usage:

```
scala> val letters = List('a', 'b', 'c')
scala> for (i <- 0 until letters.length) println(dropIndex(letters, i))
List(b, c)
List(a, c)
List(a, b)
```

Share Improve this answer Follow edited Nov 16 '13 at 10:18 answered Nov 16 '13 at 10:00

simleo
1,875 17 20

0

Using a for comprehension on a list `xs` like this,

```
for (i <- 0 until xs.size if i != nth-1) yield xs(i)
```

Also consider a set of exclusion indices, for instance `val excl = Set(2,4)` for excluding the second and fourth items; hence we collect those items whose indices do not belong to the exclusion set, namely

```
for (i <- 0 until xs.size if !excl(i)) yield xs(i)
```

Share Improve this answer Follow

answered Aug 23 '16 at 18:43

elm

18.7k 11 59 106