Insert a new element in a specified position of a list

Asked 6 years, 1 month ago Active 2 years, 4 months ago Viewed 14k times

12

6

There is no built-in function or a method of a List that would allow user to add a new element in a certain position of a List. I've wrote a function that does this but I'm not sure that its a good idea to do it this way, even though it works perfectly well:

```
def insert(list: List[Any], i: Int, value: Any) = {
  list.take(i) ++ List(value) ++ list.drop(i)
}
```

Usage:

```
scala> insert(List(1,2,3,5), 3, 4)
res62: List[Any] = List(1, 2, 3, 4, 5)
```

scala

Share Improve this question Follow

edited Mar 12 '19 at 19:15

Xavier Guihot
34.5k 16 206 129

asked Jun 24 '15 at 21:28

HeeL
285 1 2 8

2 Not an answer, but I would suggest using `.splitAt` rather than `.take` and `.drop` (to avoid going over the list twice). – Marth Jun 24 '15 at 21:35

3 This would probably be much more appropriate at CodeReview.SE. I am going to answer in the spirit of that site. – Jörg W Mittag Jun 24 '15 at 21:40

2 "There is no built-in function or a method of a List that would allow user to add a new element" But there is such a method - it is called `patch` . – Suma Dec 17 '18 at 12:10

3 Answers

Active | Oldest | Votes

39

Type Safety

The most glaring thing I see is the lack of type safety / loss of type information. I would make the method generic in the list's element type:

```
def insert[T](list: List[T], i: Int, value: T) = {
  list.take(i) ++ List(value) ++ list.drop(i)
}
```

Style

If the body only consists of a single expression, there is no need for curly braces:

```
def insert[T](list: List[T], i: Int, value: T) =
  list.take(i) ++ List(value) ++ list.drop(i)
```

Efficiency

@Marth's comment about using `List.splitAt` to avoid traversing the list twice is also a good one:

```
def insert[T](list: List[T], i: Int, value: T) = {
  val (front, back) = list.splitAt(i)
  front ++ List(value) ++ back
}
```

Interface

It would probably be convenient to be able to insert more than one value at a time:

```
def insert[T](list: List[T], i: Int, values: T*) = {
  val (front, back) = list.splitAt(i)
  front ++ values ++ back
}
```

Interface, take 2

You could make this an extension method of `List`:

```
implicit class ListWithInsert[T](val list: List[T]) extends AnyVal {
  def insert(i: Int, values: T*) = {
    val (front, back) = list.splitAt(i)
    front ++ values ++ back
  }
}

List(1, 2, 3, 6).insert(3, 4, 5)
// => List(1, 2, 3, 4, 5, 6)
```

Closing remarks

Note, however, that inserting into the middle of the list is just not a good fit for a cons list. You'd be much better off with a (mutable) linked list or a dynamic array instead.

edited May 23 '17 at 12:02
Community ♦

answered Jun 24 '15 at 22:01
Jörg W Mittag

1 Great answer. Minor nitpick : you could replace `front ++ List(values:_*) ++ back` with `front ++ values ++ back` . – Marth Jun 24 '15 at 22:05

Yep, thanks. I was too focused on keeping close to the original. – Jörg W Mittag Jun 24 '15 at 22:07

Little off top: why `ListWithInsert extends AnyVal` ? – dmitry Jun 24 '15 at 23:49

@dmitry: Why not? After all, enrich-my-library was one of the primary motivators for introducing value classes in the first place. – Jörg W Mittag Jun 25 '15 at 1:18

Thank you, I haven't realize that such a service for code reviews exists. I will definitely use it in the nearest future with some more scala examples – HeeL Jun 26 '15 at 12:50

32

You can also use `xs.patch(i, ys, r)` , which replaces `r` elements of `xs` starting with `i` by the patch `ys` , by using `r=0` and by making `ys` a singleton:

```
List(1, 2, 3, 5).patch(3, List(4), 0)
```

Share Improve this answer Follow

answered Jun 24 '15 at 23:30

Ben Reich
15.6k 2 34 55

4

In [the Scala course by his eminence Martin Odersky himself, he implements it similarly to]

```
def insert(list: List[Any], i: Int, value: Any): List[Any] = list match {
  case head :: tail if i > 0 => head :: insert(tail, i-1, value)
  case _ => value :: list
}
```

One traversal at most.

Share Improve this answer Follow

answered Mar 8 '19 at 11:07

serv-inc
30k 9 131 149