Unit 5d: Folding

1 Introduction

Now let's return to function composition and combinators using an example which is curried. Here is a problem we can solve using a different functional combinator - add together all the elements of a list of integers.

The imperative solution would involve the following steps:

- Declare an accumulator variable
- Iterate through collection
- For each item in collection, add its value to the accumulator variable
- When iteration is complete, return value of accumulator variable

Again, the functional solution notes that there are two transforms here:

- Transform two values to a single value by adding them
- Transform the collection to a value by applying the first transform in turn to each element and the result of applying it to previous elements

The second transform is known as folding.

Using the same list as before, we can add the values using the *foldLeft* function as follows:

```
List(1,2,3,4,5,6).foldLeft(0)((x,y) => x + y)
```

The addition function is defined as lambda where the first parameter of the (curried) function is an accumulator, so the first addition will add 0 to the first element.

Here we show what is going on more clearly by printing the values of x and y each time the addition is evaluated:

```
scala> List(1,2,3,4,5,6).foldLeft(0){(x,y)=>println("x:
"+ x + " y: "+y);x + y}
x: 0 y: 1
x: 1 y: 2
x: 3 y: 3
x: 6 y: 4
x: 10 y: 5
x: 15 y: 6
res30: Int = 21
```

For each element, y is the current element and x is the accumulated result from the previous elements (what do you think foldRight would do? – you should try it and see).

2 Folding with other functions

As you saw previously, function composition is very flexible. Folding can be used to aggregate the elements in a collection in different ways, by composing with any function that combines two values into one. For example, we could multiply together instead (with accumulator set to 1):

```
List(1,2,3,4,5,6).foldLeft(1)((x,y) => x * y)
```

Note that this example calculates the factorial of 6 – an alternative to using recursion.

Folding is often expressed with a syntax shorthand – another use for that underscore symbol, replacing a lambda that takes two parameters as follows:

```
List(1,2,3,4,5,6).foldLeft(0)(_ + _)
List(1,2,3,4,5,6).foldLeft(1)(_ * _)
```

Other function combinators include *foreach* and *map* – see Scaladocs for collection classes.

3 foldLeft as an example of control abstraction

We can put the second parameter of *foldLeft* in {}, like this:

```
List(1,2,3,4,5,6).foldLeft(0){
  (x, y) => x + y
}
```

foldLeft is then being used as a control abstraction; this reads like a control structure where the second parameter, which defines the function to be applied while folding, is written in {} like a code block.

foldLeft is a function defined as a method of the *List* class – even though many types have a method of this name, it is not actually built into the Scala language.

4 foldRight

From the Scala documentation:

```
def foldRight[B](z: B)(op: (A, B) ⇒ B): B
```

Applies a binary operator to all elements of this list and a start value, going right to left.

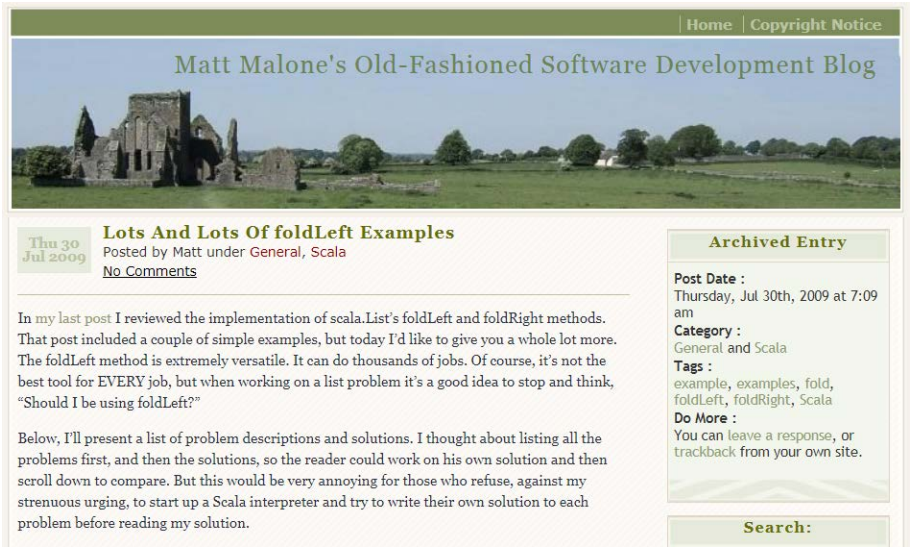
B	the result type of the binary operator.
z	the start value.
op	the binary operator.
returns	the result of inserting op between consecutive elements of this list, going right to left with the start value z on the right:

```
op(x_1, op(x_2, ... op(x_n, z)...))
```

where $x_1, \ldots, x_n$ are the elements of this list. Returns z if this list is empty.

Definition Classes List → LinearSeqOptimized → IterableLike → TraversableOnce → GenTraversableOnce

5 Fold Examples



Home | Copyright Notice

Matt Malone's Old-Fashioned Software Development Blog

Thu 30 Jul 2009

Lots And Lots Of foldLeft Examples

Posted by Matt under General, Scala

No Comments

In my last post I reviewed the implementation of scala.List's foldLeft and foldRight methods. That post included a couple of simple examples, but today I'd like to give you a whole lot more. The foldLeft method is extremely versatile. It can do thousands of jobs. Of course, it's not the best tool for EVERY job, but when working on a list problem it's a good idea to stop and think, "Should I be using foldLeft?"

Below, I'll present a list of problem descriptions and solutions. I thought about listing all the problems first, and then the solutions, so the reader could work on his own solution and then scroll down to compare. But this would be very annoying for those who refuse, against my strenuous urging, to start up a Scala interpreter and try to write their own solution to each problem before reading my solution.

Archived Entry

Post Date :
Thursday, Jul 30th, 2009 at 7:09 am
Category :
General and Scala
Tags :
example, examples, fold, foldLeft, foldRight, Scala
Do More :
You can leave a response, or trackback from your own site.

Search:

https://oldfashionedsoftware.com/2009/07/30/lots-and-lots-of-foldleft-examples/