

One-liners are your friend: increasing productivity with Scala

thomas.alexandre@devcode.se








One-liners are your friend

➤ **Less code shouldn't decrease readability:**

- `List(1,2,3,4,5).filter { theCurrentNumber => theCurrentNumber < 4 }`
- `List(1,2,3,4,5).filter { number => number < 4 }`
- `List(1,2,3,4,5).filter { n => n < 4 }`
- `List(1,2,3,4,5).filter { _ < 4 }`

➤ **Our definition: short statements, possibly on 2 lines 😊**

Possible setup for Enterprise Development

	JAVA	SCALA
Domain Model	POJOs	Case classes
REST API		
- Server	 	 
- Client	HttpClient, ...	Dispatch, ...
Asynchronous, concurrent	Futures, Threads and Locks	 Futures Dataflow concurrency

COUPONS Service

GET

http://localhost:8086/deals

```
<?xml version="1.0">
<deals>
  <deal>
    <title> Helicopter </title>
    <discount> 60 </discount>
    <city> Stockholm </city>
  </deal>
  ...
</deals>
```

The screenshot shows the Groupon website interface. At the top, a navigation bar lists various Swedish cities: Stockholm, Göteborg, Malmö, Uppsala, and Linköping. Below this, a grid of cities is displayed, including Södertälje, Lund, Borlänge, Eskilstuna, Falun, Halmstad, Helsingborg, Jönköping, Kalmar, Karlskrona, Kristianstad, Luleå, Norrköping, Skövde, Sundsvall, Umeå, Visby, Västerås, Växjö, and Örebro. A 'Shopping' and 'Getaways' section is also visible. The Groupon logo is prominently displayed in the center. To the right, a 'Dagens deal:' dropdown menu is open, showing 'Stockholm' as the selected city. Below the logo, there are navigation links for 'Aktuella deals', 'Tidigare deals', and 'Så fungerar det'. A social media bar includes 'Bjud in dina vänner' and links to Facebook, Twitter, and Mail. The main content area features a deal for a 'Guidad helikoptertur över Stockholm och skärgården. Värde 2 500 kr, betala 999 kr.' with a green 'Köp nu!' button. Below the button, the total price is 'Summa: 999,00 kr'. A red box highlights a 'Rabatt 60%' section, and another red box highlights the 'Dagens deal:' dropdown menu. Below the price, there is a 'Köp till en vän!' button and a section titled 'Detta erbjudande kan fortfarande köpas inom:' with a countdown timer showing '08 27 36'.

REPL (Read-Eval-Print Loop)

```
scala> case class Deal( title:String= "Empty", discount:Int = 0, city:String = "Stockholm")
defined class Deal
```

```
scala> █
```

```
1  package se.devcode.jfokus;
2
3  public class Deal {
4
5      final String title;
6      final Integer discount;
7      final String city;
8
9      public Deal(String title, Integer discount, String city) {
10         super();
11         this.title = title == null ? "Empty" : title;
12         this.discount = discount == null ? 0 : discount;
13         this.city = city == null ? "Stockholm" : city;
14     }
15
16     public String getTitle() {
17         return title;
18     }
19
20     public Integer getDiscount() {
21         return discount;
22     }
23
24     public String getCity() {
25         return city;
26     }
27
28     @Override
29     public String toString() {
30         return "Deal [title=" + title + ", discount=" + discount + ", city="
31             + city + "]";
32     }
33
34     // TODO: create hashCode(), equals() ...
35 }
```

Summary of Map/Reduce as a runnable program...

```
1 package se.devcode.jfokus
2
3 import dispatch._
4
5 case class Deal(title:String = "Empty", discount:Int = 0, city:String = "Stockholm" )
6
7 object RestClient {
8
9     def main(args: Array[String]) {
10
11         val request = url("http://localhost:8086/deals")
12
13         val dealsAsXml = Http( request <> { r => r \ "deal" } )
14
15         println("All Deals as XML: "+ dealsAsXml)
16
17         val deals = dealsAsXml map { d => Deal((d \ "title").text,(d \ "discount").text.toInt,(d \ "city").text)}
18
19         val (badDeals,goodDeals) = deals partition { _.discount < 20 }
20
21         val goodDealsByCity = goodDeals groupBy { _.city }
22
23         val nbOfGoodDealsByCity = goodDealsByCity mapValues { _.size }
24
25         println("Good Deals sorted by city: " + nbOfGoodDealsByCity)
26
27     }
28
29 }
30
```

... as a Tweetable Oneliner



T @scalaoneliner

2m

```
(for{x<-Http(r<>[_\"deal\"])};d=(x\"disc\").text.toInt if d>20}yield  
Deal((x\"title\").text,d,(x\"city\").text))groupBy(_.city)mapValues(_.size)
```



Akka dataflow concurrency

```
1 package se.devcode.jfokus
2
3 import dispatch._
4 import akka.dispatch._
5 import Future.flow
6
7 object AkkaSample {
8
9   def main(args: Array[String]) {
10
11     val params = Promise[Map[String,String]]()
12
13     val deals = Future( Http( url("http://localhost:8086/deals") <<? params.get as_str ))
14
15     println ("Result: "+ deals.result)
16
17     flow { params << Map( "city" -> "stockholm" ) }
18
19     Thread.sleep(2000)
20
21     println("Result: "+ deals.result)
22
23   }
24 }
```


Additional Links

- **Dispatch:** <http://dispatch.databinder.net/Dispatch.html>
- **Akka:** <http://www.typesafe.com/technology/akka>
- **Videos online from Jfokus 2009-2012, ScalaDays, Scala eXchange, NEScala...**
- **InfoQ article (by me): “An Introduction to Scala for Java Developers”**
<http://www.infoq.com/articles/scala-for-java-devs>
- **Fast Track to Scala Course @DevCode, Stockholm**





**Do You Have
Any Questions?**



Extra: Testing with Unfiltered

```
object RestClientSpec extends SpecificationWithJUnit
    with unfiltered.spec.jetty.Served {

  def setup = { _.filter(unfiltered.filter.Planify {
    case _ => Xml(<deals>
      <deal>
        <title>Helicopter Tour</title>
        <discount>60</discount>
        <city>Stockholm</city>
      </deal>
      <deal>
        <title>10% OFF on Hotel Bedroom</title>
        <discount>10</discount>
        <city>Paris</city>
      </deal>
    </deals>))})

  "Getting deals with more than 20% discount " should {

    "return only one Stockholm deal" in {
      val restClient = new RestClient
      restClient.nbOfGoodDealsByCity(host) must_== Map("Stockholm" -> 1)
    }
  }
}
```