

thomas.alexandre@devcode.se





One-liners are your friend

- Less code shouldn't decrease readibility:
 - List(1,2,3,4,5).filter { theCurrentNumber => theCurrentNumber < 4 }</p>
 - List(1,2,3,4,5).filter { number => number < 4 }</p>
 - List(1,2,3,4,5).filter { n => n < 4 }</p>
 - List(1,2,3,4,5).filter { _ < 4 }</p>

Our definition: short statements, possibly on 2 lines ©



Possible setup for Enterprise Development

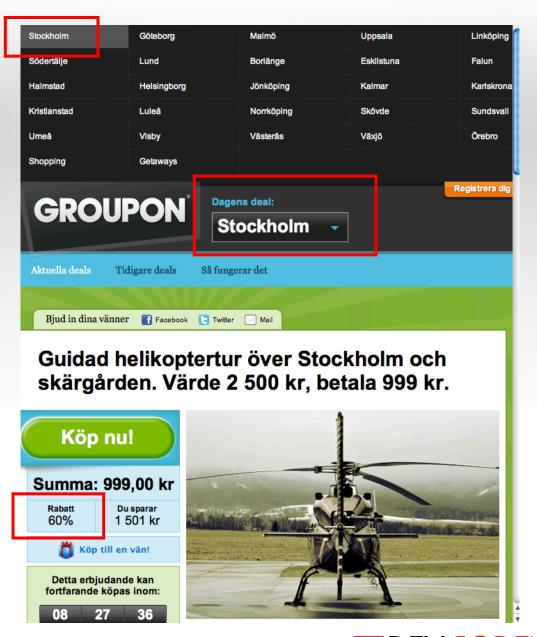
	JAVA	SCALA
Domain Model	POJOs	Case classes
REST API		
- Server	springroo Play!	unfiltered <i>Lift</i>
- Client	HttpClient,	Dispatch,
Asynchronous, concurrent	Futures, Threads and Locks	akka Futures Dataflow concurrency



COUPONS Service

GET

http://localhost:8086/deals





REPL (Read-Eval-Print Loop)

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

scala>

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

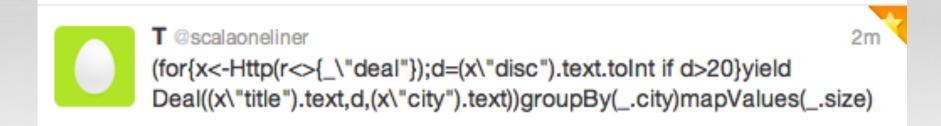


Summary of Map/Reduce as a runnable program...

```
package se.devcode.jfokus
 2
 3
    import dispatch._
 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
       val dealsAsXml = Http( request <> { r => r \\ "deal" } )
13
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
       val (badDeals, goodDeals) = deals partition { _.discount < 20 }</pre>
19
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







Akka dataflow concurrency

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

