AUTO SCOUT 24

Combine it!
(the functional way)

www.scout24.com

Munich | 12.10.2018 | Martin Lechner

Task: Group this dataset by keys and sum values

```
List(Map("A" → 1, "B" → 2), Map("A" → 41))
```



```
Map("A" → 42, "B" → 2)
```

AUTO SCOUT 24

Manually doing this may look like

```scala
def manualCombine(): Map[String, Int] = {
  val initialMap = MMap.empty[String, Int].withDefaultValue(0)
  data
    .foldLeft(initialMap) {
      (finalMap: MMap[String, Int], rowMap: Map[String, Int]) ⇒
        rowMap.foreach { case (key, value) ⇒ finalMap(key) += value }
        finalMap
    }
    .toMap
}
```

The cats way

```
def catsCombine(): Map[String, Int] = data.combineAll
```

WTF – how is this working ?!?

```
trait Combineable[A] {
  def combine(x: A, y: A): A
}
```

WTF – how is this working ?!?

```
trait Combineable[A] {
  def combine(x: A, y: A): A
}
```

```
val CombinableInt = new Combineable[Int] {
  override def combine(x: Int, y: Int): Int = x + y
}
```

WTF – how is this working ?!?

```
trait Combineable[A] {
  def combine(x: A, y: A): A
}
```

```
val CombinableInt = new Combineable[Int] {
  override def combine(x: Int, y: Int): Int = x + y
}
```

- For folds / reduce it might be useful to have an empty element:

```
trait CombineableWithEmpty[A] extends Combineable[A] {
  def empty: A
}
```

The missing piece: Typeclasses

- Typeclasses encapsulate / abstract behavior away from specific type
- Usually done with implicits in Scala
- You can add the Typeclass behavior by providing typeclass instances
- Worth a talk on their own...

Typeclass example

```scala
trait CombineableTypeclass[A] {
  def combine(x: A, y: A): A
}

object CombinableTypeclass {

  def combine[A](a: A, b: A)(implicit c: CombineableTypeclass[A]) =
    c.combine(a, b)

  implicit val CombinableInt = new CombineableTypeclass[Int] {
    override def combine(x: Int, y: Int): Int = x + y
  }

}
```

Meanwhile in cats world

- Combineable is called Semigroup
- |+| is combine
- Rule: Associativity:
  - (a |+| b) |+| c === a |+| (b |+| c)
- CombineableWithEmpty is called Monoid
- Additional rule:
  - a |+| empty === empty |+| a === a
- Cats provides default instances for those typeclasses!
- List, Map, String, Int, ... all covered for free!

Full example with custom ADT

- Show code in Intellij

Thank you

- Register for the workshop with Luka Jacobowitz, core Maintainer of Cats!