

Débuter en Scala :

entre incompréhension et
for-compréhension

Mon parcours

2016 : Diplôme en informatique et réseaux

2017 : Développeuse Back-End à Tabmo



Programmation orientée objet

Programmation fonctionnelle



Sommaire

Sca-quoi ?
Découvrir le sujet
Moult ressources
Et avec ça ?
Bye Bye les habitudes Java
De riches trouvailles
What else ?
A bright future

Sca-quoi ? Perception du Scala par le néophyte

| SCALA |
is
the
new JAVA |

FIFTY SHADES
OF SCALA

Sca-quoi ? Perception du Scala par le néophyte



Découvrir le sujet -

Un typage systématique - **inférence** de type

Scala

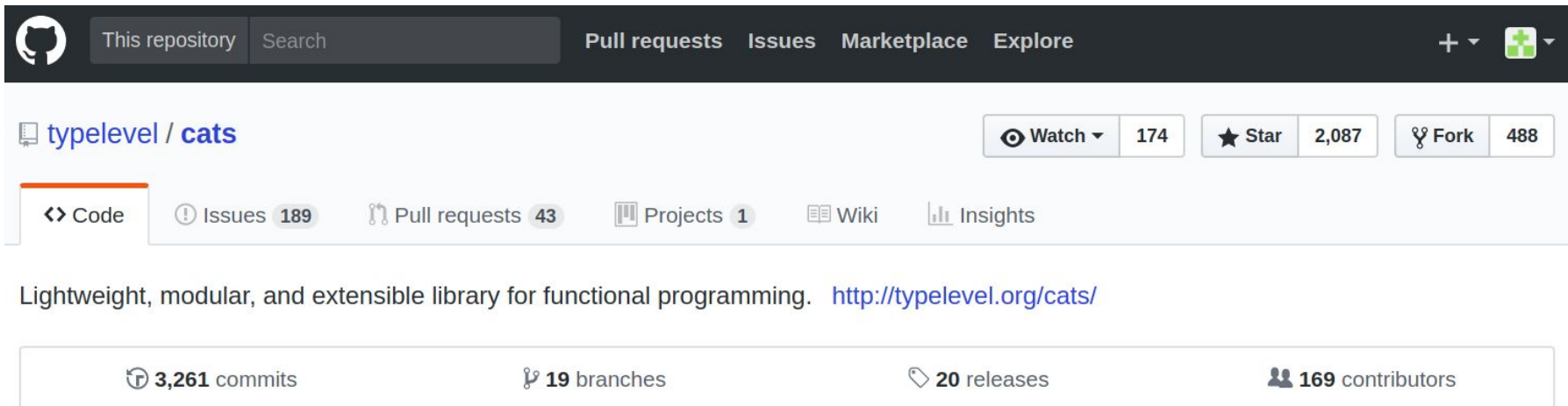
```
val myMap = Map(1 -> "un pacito palante Maria")  
  
// Map[Int, String]
```

Java

```
Map<Integer, String> myMap = new HashMap<Integer, String>()
```

Découvrir le sujet

Une communauté très active, des librairies et des frameworks en



The screenshot shows the GitHub interface for the 'typelevel / cats' repository. At the top, there's a dark navigation bar with the GitHub logo, a search bar, and links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below this, the repository name 'typelevel / cats' is displayed. To the right of the name are buttons for 'Watch' (174), 'Star' (2,087), and 'Fork' (488). A secondary navigation bar contains links for 'Code', 'Issues' (189), 'Pull requests' (43), 'Projects' (1), 'Wiki', and 'Insights'. The repository description reads: 'Lightweight, modular, and extensible library for functional programming.' followed by the URL 'http://typelevel.org/cats/'. At the bottom, a summary bar shows '3,261 commits', '19 branches', '20 releases', and '169 contributors'.

typelevel / cats

Watch 174 Star 2,087 Fork 488

Code Issues 189 Pull requests 43 Projects 1 Wiki Insights

Lightweight, modular, and extensible library for functional programming. <http://typelevel.org/cats/>

3,261 commits 19 branches 20 releases 169 contributors

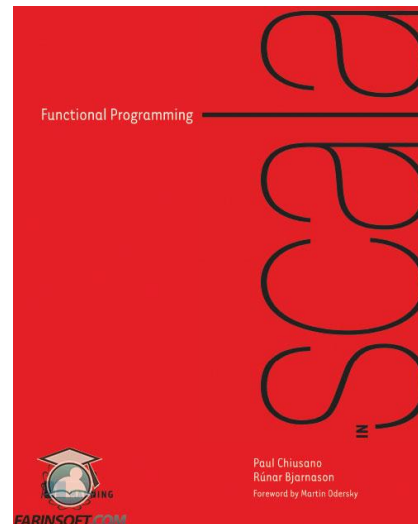
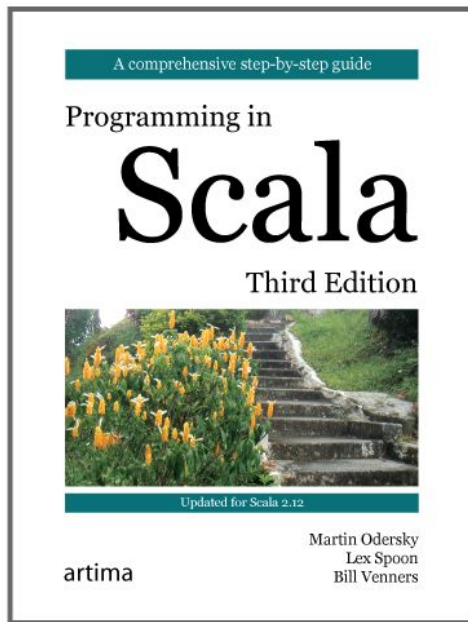
Applications **concurrentes** et **distribuées**

Moult ressources - Outils

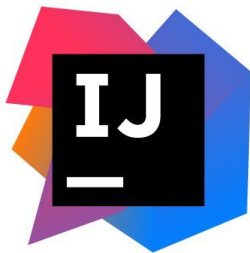


coursera

Daniel Westheide
on making software



Moult ressources - Outils



```
melanie@tabmo-melanie:~$ scala
Welcome to Scala 2.11.8 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_111).
Type in expressions for evaluation. Or try :help.

scala> Map(1 -> Seq("un", "dos", "tres"), 2 -> Seq("un", "pasito", "palante"))
res0: scala.collection.immutable.Map[Int,Seq[String]] = Map(1 -> List(un, dos, tres),
```

Et avec ça ? Du Scala, oui mais pas seul(e)

Sur des projets existants

Mentoring



Bye Bye les habitudes Java

A la recherche de notions perdues



Des boucles itératives ?

Des return ?

Des 'new' ?

Bye Bye les habitudes Java

A la recherche de notions perdues



Des valeurs mutables ?

```
case class Employee(name: String, office: String, role: String)
```

```
val fred = Employee("Fred", "Anchorage", "Salesman")
```

```
val joe = fred.copy(name = "Joe")
```

De riches trouvailles - De 'nouvelles' notions



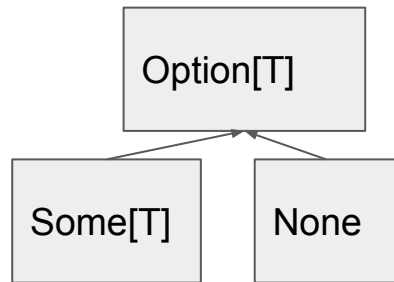
les traits

les options, les either, les futures

le pattern matching

De riches trouvailles - Les options

Gestion des cas limites



Fonction avec un identifiant unique retour d'une valeur unique

```
def getOrder(orderId: OrderId): Option[Order]
```

```
val orderOption = getOrder("4512-6545")
```

```
if (orderOption.isDefined) orderOption.get.toString  
else "OrderId is not found."
```

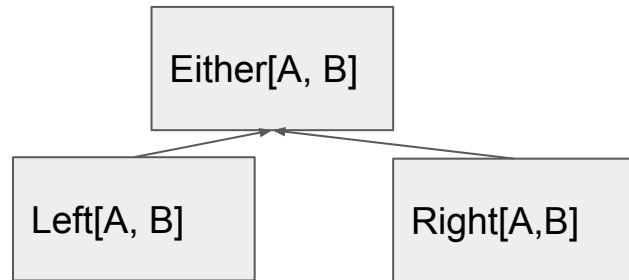
```
1) orderOption.map(order => order.toString).getOrElse(  
  "No order Found")
```

```
2) orderOption.map(order => fetch(order)).foreach(println)
```

De riches trouvailles - les Either

Détails sur l'erreur

Collecter les erreurs



```
def divide(x: Double, y: Double): Either[String, Double]
```

```
val result = divide(4, 2) // Right(2.0)
```

```
if (result.isRight) result.right.get  
else result.left.get
```

```
result.fold({error => s"An Error occured:$error" },  
{res => " Result is $res"})
```

Des trouvailles - Les Futures et les Promises

Execution Context Global / Local

Pool de threads

Callback

Acteurs

- .onComplete
- .onSuccess
- .onFailure

SCHRÖDINGER'S CAT IS
ALIVE



Composables

De riches trouvailles

- **Le pattern Matching**

```
def fact(n: Int): Int = n match {  
  case 0 => 1  
  case n => n * fact(n-1)  
}
```

```
def canFly(animal: Animal): Boolean = animal match {  
  case bird: Bird => true  
  case animal if animal.hasSuperPower => true  
  case _ => false  
}
```

```
def length[T](list: List[T]): Int = list match {  
  case Nil => 0  
  case _ :: tail => 1 + length(tail)  
}
```

De riches trouvailles - for-compréhension

```
(1 until n).flatMap(i =>
  (1 until i).filter(j => isPrime(i + j)).map
    (j => (i, j)))
```

équivalent à

```
for {
  i <- 1 until n
  j <- 1 until i
  if isPrime(i + j)
} yield (i, j)
```



De riches trouvailles - for-compréhension

```
for {  
  i ← A  
  j ← B  
  k ← C  
} yield (i, j, k)
```



```
A: Future[Seq[String]]  
B:   DBIO[Seq[String]]  
C: Option[Seq[String]]
```

De riches trouvailles - Les fonctions

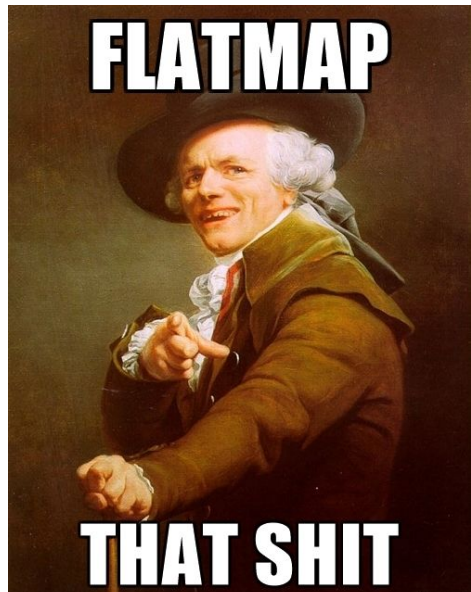


les fonctions déclarées dans des fonctions

les fonctions passées en paramètre de fonctions

les fonctions retournées par des fonctions

De riches trouvailles - Les life savers



```
Future[Seq[Seq[String]]]  
Future[Seq[Future[Either[ApiError, Json]]]]
```



```
Future.sequence  
Seq[Future[A]] => Future[Seq[A]]
```

```
Future.successful  
A => Future[A]
```

```
Future[Future[String]]
```

```
future.map(Right.apply)  
Future[A] => Future[Either[?, A]]
```

A bright future - Conclusion

Encore moult librairies à maîtriser
Dobbie (DB), Cats (abstractions), Circe (JSON),
Shapeless (Generic Programming)

Creuser des concepts : Free Monades & co'

Progresser dans la matrice



Level A1: Beginning application programmer

- Java-like statements and expressions: standard operators, method calls, conditionals, loops, try/catch
- class, object, def, val, var, import, package
- Infix notation for method calls
- Simple closures
- Collections with map, filter, etc
- for-expressions

Level A2: Intermediate application programmer

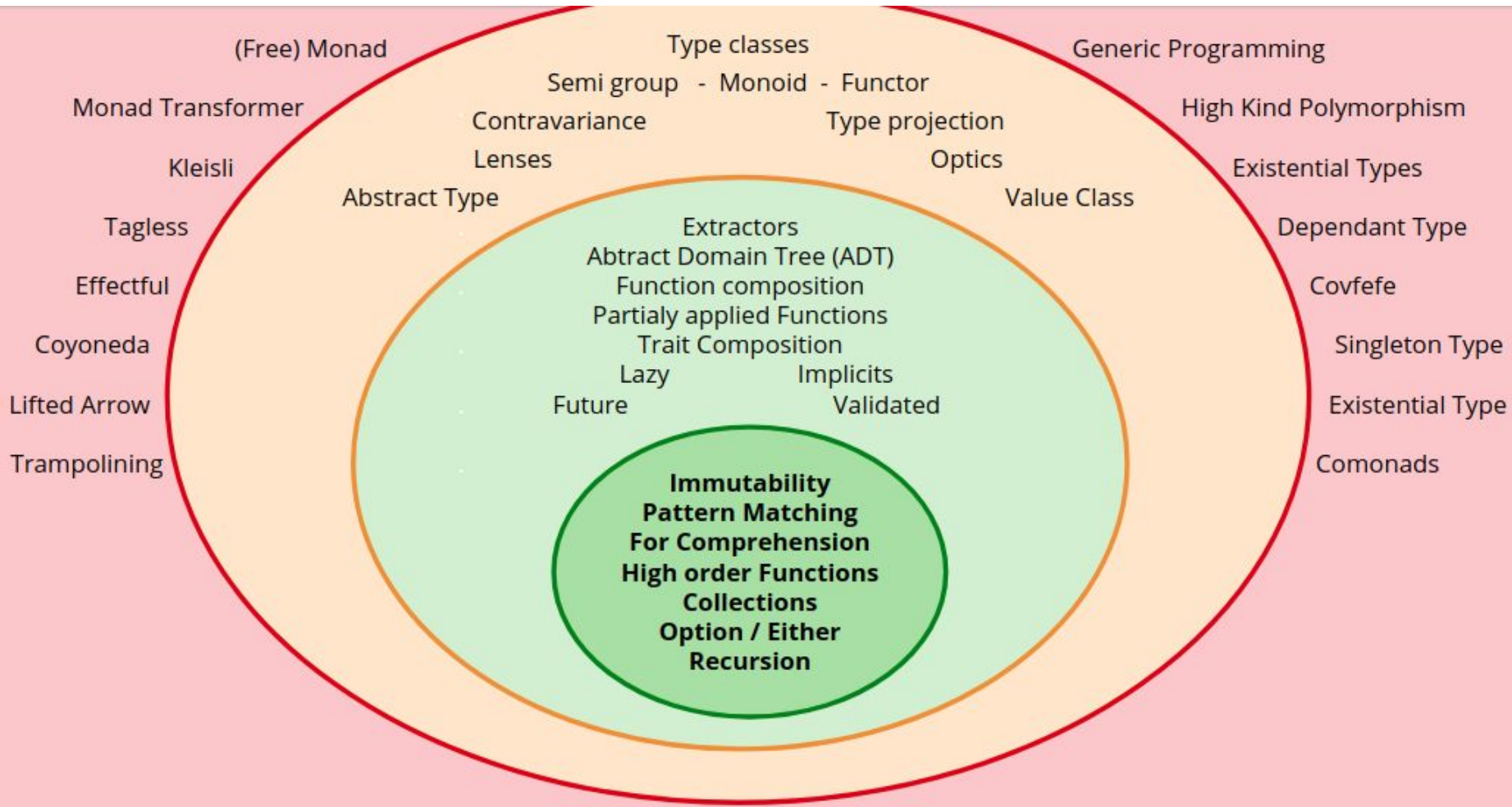
- Pattern matching
- Trait composition
- Recursion, in particular tail recursion
- XML literals

Level L1: Junior library designer

- Type parameters
- Traits
- Lazy vals
- Control abstraction, currying
- By-name parameters

Level L2: Senior library designer

- Variance annotations
- Existential types (e.g., to interface with Java wildcards)
- Self type annotations and the cake pattern for dependency injection
- Structural types (*aka* static duck typing)
- Defining map/flatMap/withFilter for new kinds of for-expressions



A bright future - Conclusion

Des moyens d'apprentissage plus ludiques en préparation



A close-up photograph of Bear Grylls. He is looking directly at the camera with a serious expression. His face is covered in mud, particularly around his eyes and nose. He is wearing a dark jacket and a blue shirt. His right hand, wearing a black glove, is extended towards the camera, with his index finger pointing. The background is a blurred green forest.

Improvise. Adapt. Overcome