Paris JUG

www.parisjug.org

13/04/2010

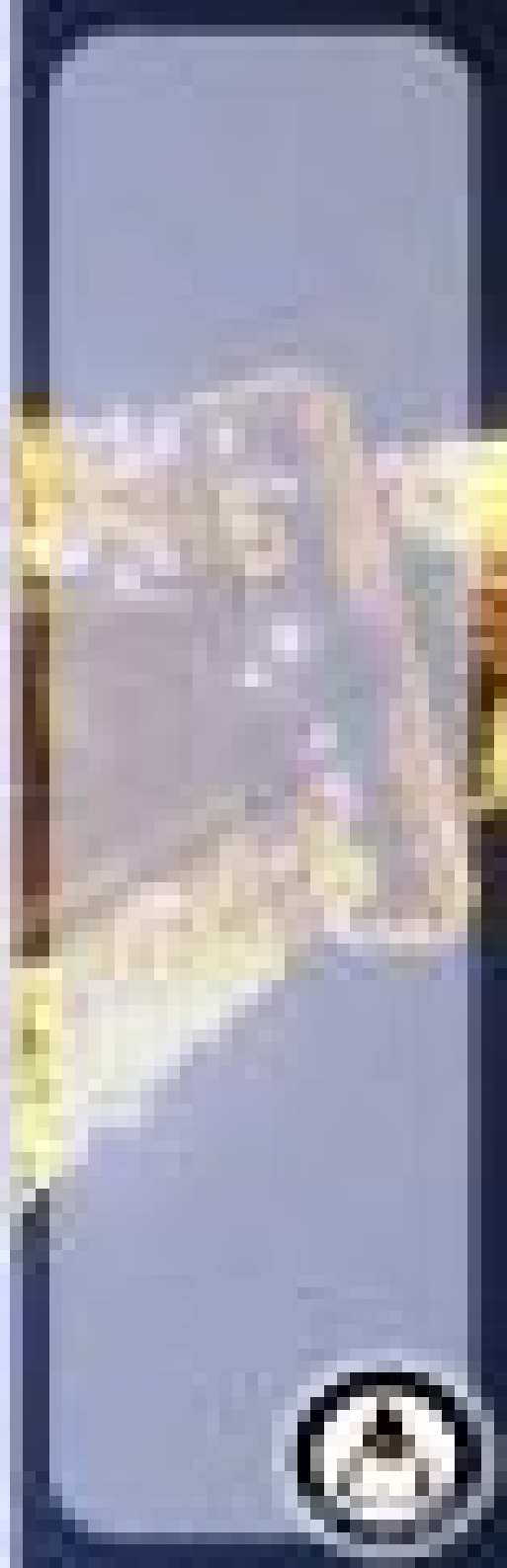
13/04/2010

Scala

Nicolas Jozwiak

Xebia

Xebia
SOFTWARE DEVELOPMENT DONE RIGHT



www.parisjug.org

13/04/2010

Scala

Romain Maton

Xebia

Twitter : rmat0n

Xebia

SOFTWARE DEVELOPMENT DONE RIGHT

www.parisjug.org

Citations

Charles Nutter (JRuby) : Scala is most likely candidate to replace Java, compared to Groovy and JRuby. While Scala is not a dynamic language, it has many of the characteristics of popular dynamic languages, through its rich and flexible type system, its sparse and clean syntax, and its marriage of functional and object paradigms.

James Strachan (Groovy) : I'm very impressed with it ! I can honestly say if someone had shown me the Programming Scala book by by Martin Odersky, Lex Spoon & Bill Venners back in 2003 I'd probably **have never created Groovy**.

James Gosling (Java) : During a meeting in the Community Corner, a participant asked an interesting question: "Which Programming Language would you use **now** on top of JVM, except Java?"

The answer was surprisingly fast and very clear : **Scala.**

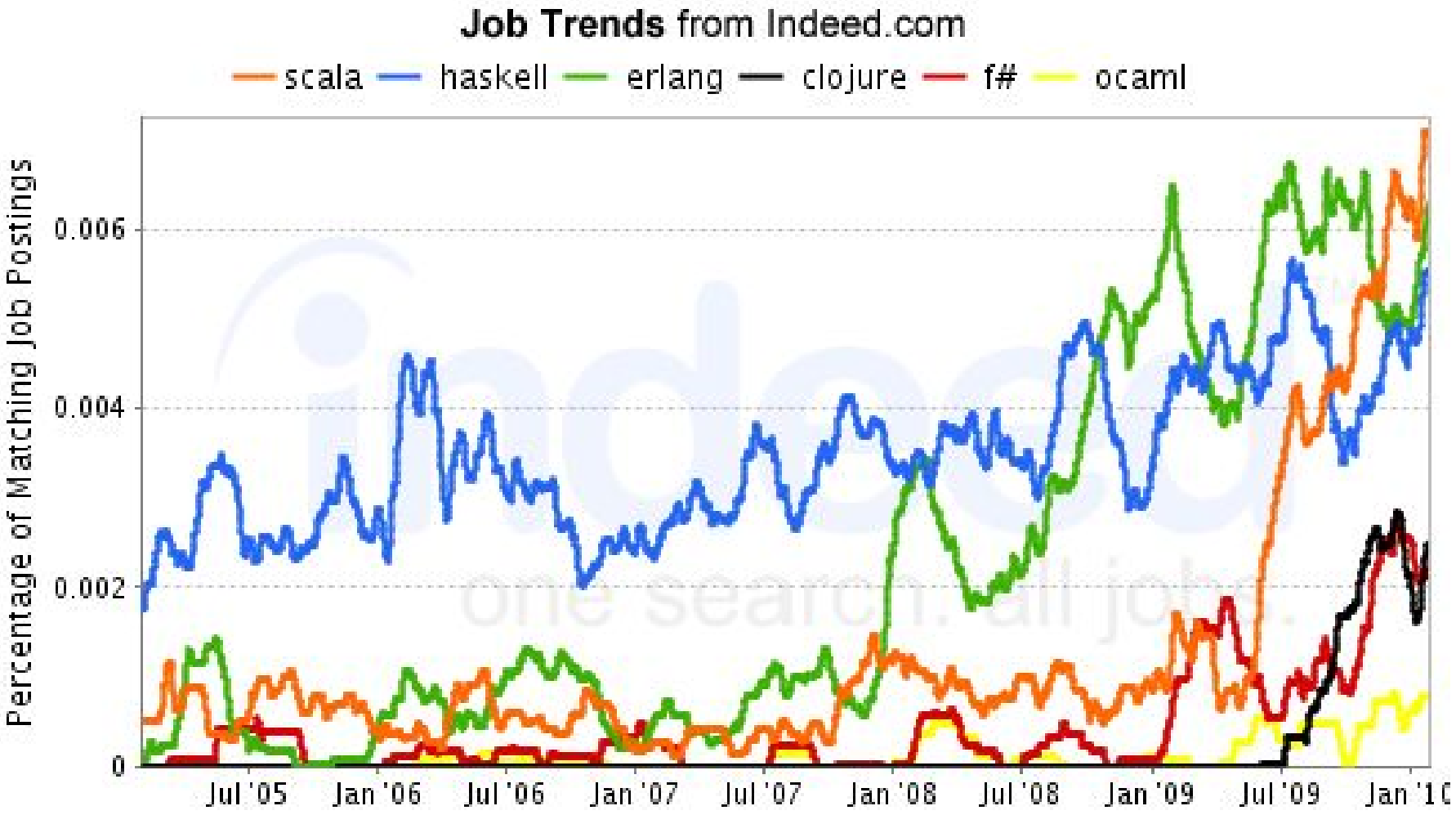
Sommaire

- Les origines de Scala ?
- Qu'y a-t-il sous le capot ? Quelques exemples
- L'outillage et les frameworks
- Pourquoi faut-il s'y intéresser ?
- Difficultés/Challenges
- Les freins à son adoption
- Le futur ?
- Qui l'utilise ?
- Conclusion

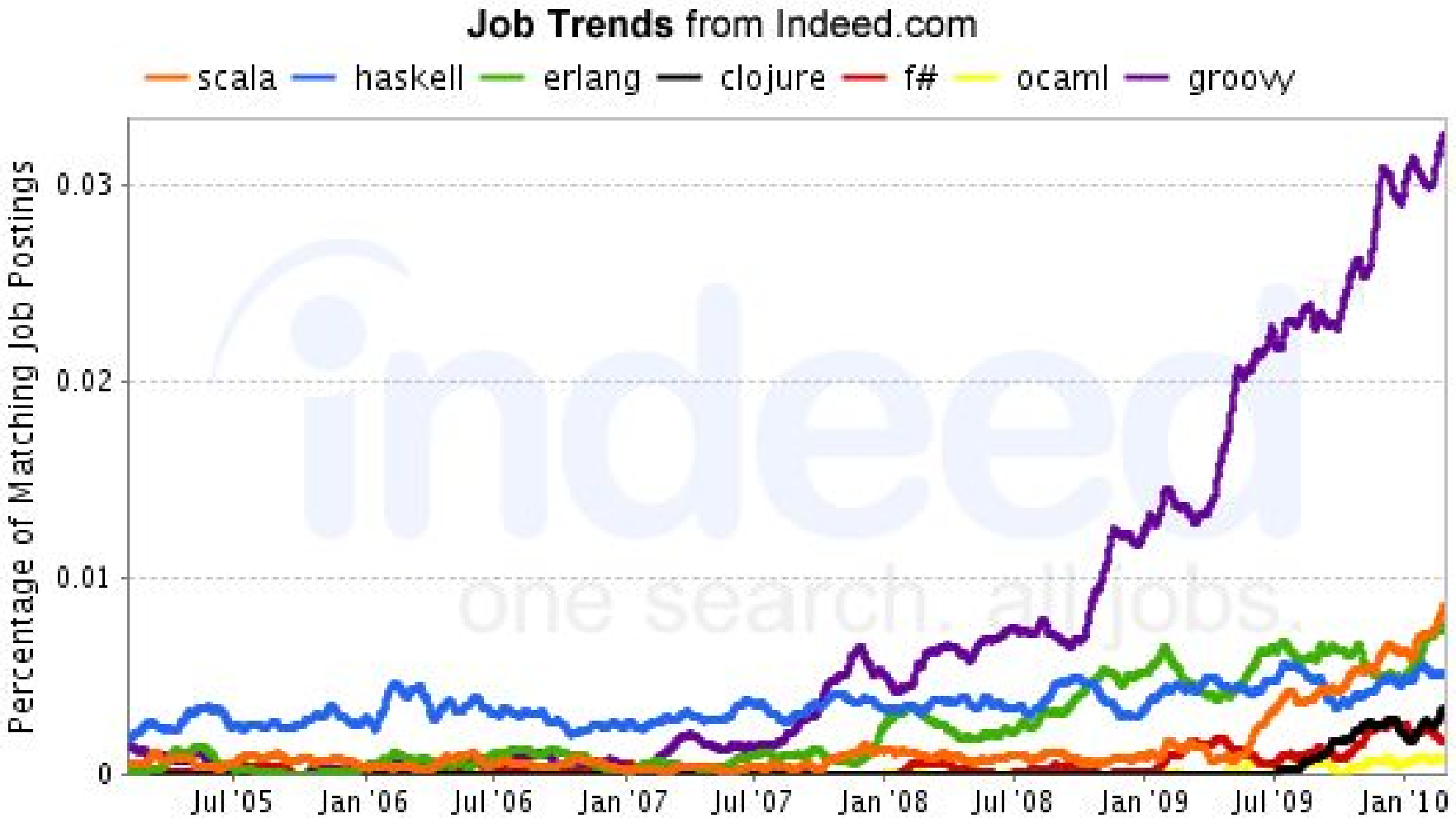
Les origines

- **Créateur : Martin Odersky**
  - Professeur de méthodes de programmation à l'EPFL (École Polytechnique Fédérale de Lausanne)

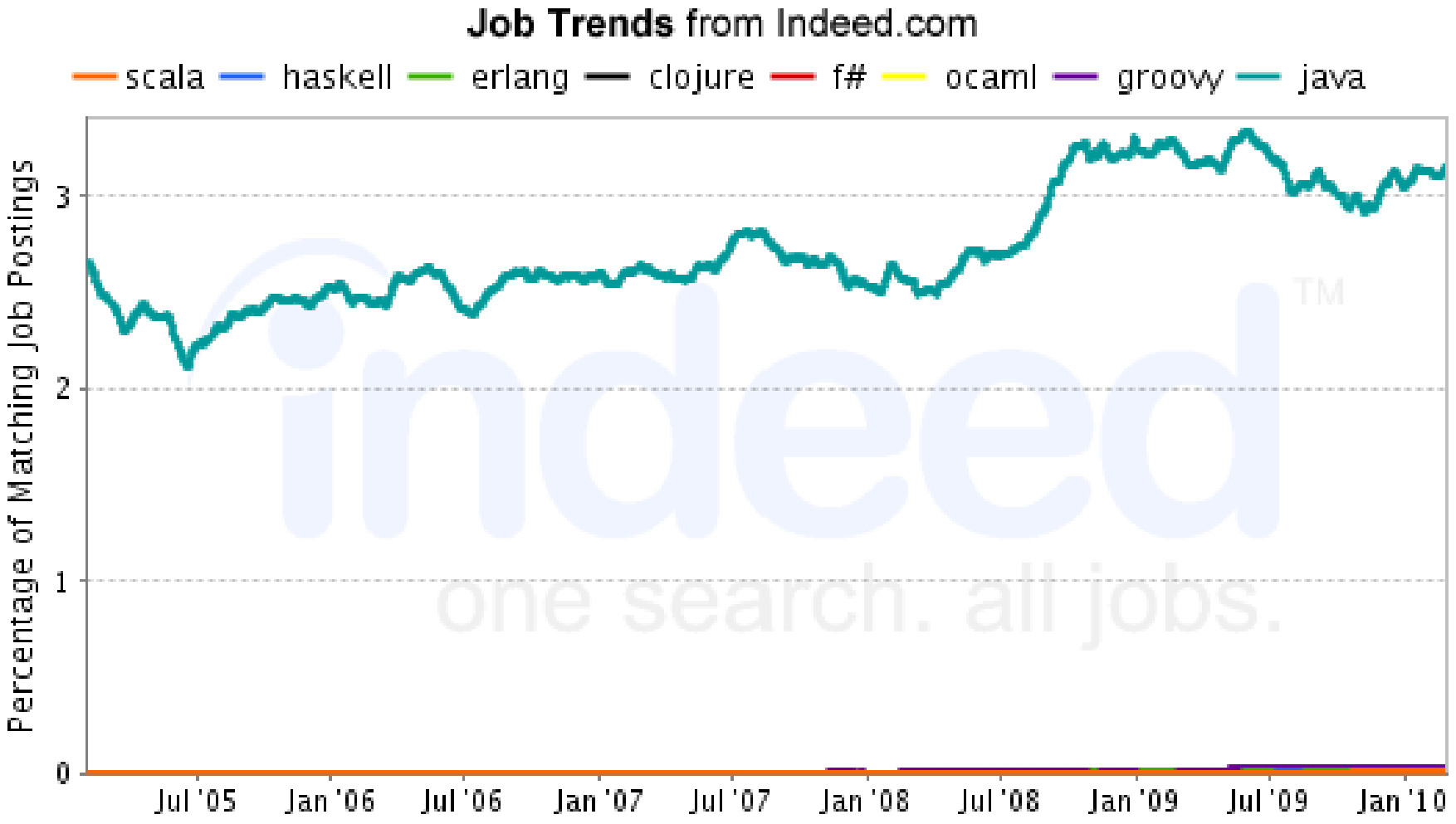
Scala... ça décolle !



Groovy loin devant



Et la route est encore longue



Scala

- **C'est un compromis entre la programmation orientée objet et la programmation fonctionnelle**
  - **Orientée Objet :** Structuration des données et richesse des APIs
  - **Fonctionelle :** Moindre verbosité et composition de fonctions
  - Scalable, programmation concurrente et parallèle
  - Scala est exécutable sur une JVM
  - 100% compatible avec Java
  - Statiquement typé

Scala

- La non nullité, Scala prônera
- L'immutabilité, Scala prônera
- Tout objet, Scala sera

1+1 <=> (1).+(1)

Quelques exemples

Inférence

```
val listStr:Array[String] = new Array[String]
val listStr = new Array[String]
```

Déclaration de fonctions

```
def multiply(x:Int, y:Int) = x*y
Multiply(2, 10)

def multiply(x:Int)(y:Int) = x*y
Multiply(2)(10)

def double = multiply(2)
double(10)
```

Quelques exemples

Listes

```
Collections.sort(...)
list.filter({x:Int => x%2==0})
list.map({x:Int => x*2})
```

Support XML natif

```
class HelloWorld {
  def hello = <span>Welcome Juggers for
     Scala's presentation ! {new java.util.Date}</span>
}
```

Passer de l'objet au fonctionnel

```
List<Integer> listInteger = new LinkedList<Integer>();
for (Integer number : list) {
      if (number % 2 == 0) listInteger.add(number);
}
```

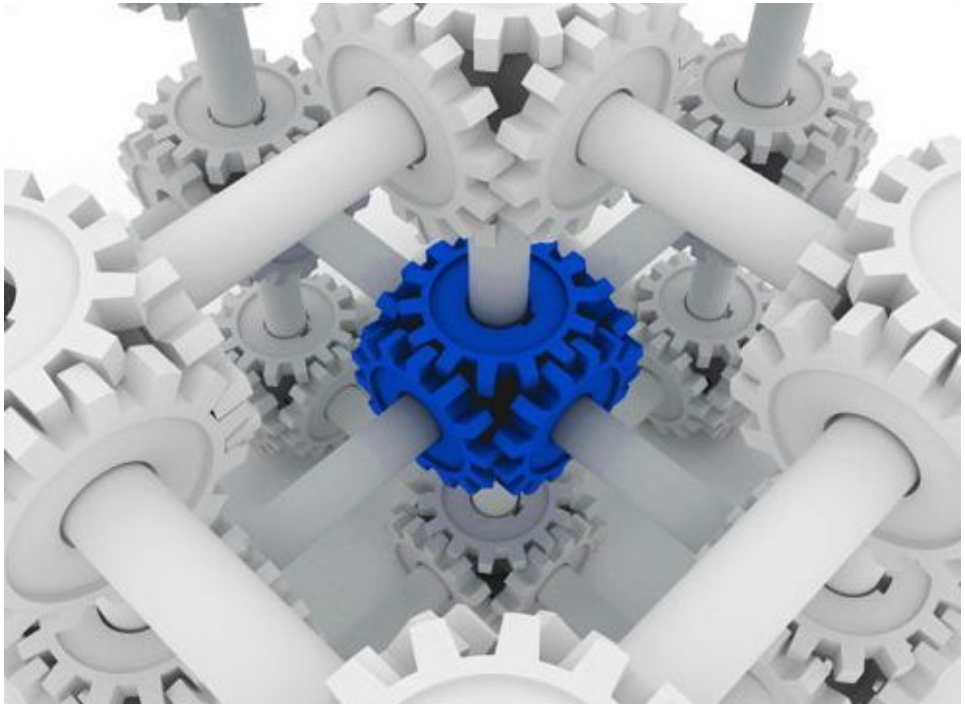
```
val listNumber = list.filter(_%2 == 0);
```

```
Predicate predicateNumber = new Predicate () {
    @Override
     public boolean evaluate (Object o) {
        Integer number = (Integer) o;
        return n % 2 == 0;
     }
}

CollectionUtils.filter(list, predicateNumber);
```

Quelques exemples

```
Round(FixingIndex("Euribor 06 Mois",
DateFixing("Prefixe", -2, 0, "Ouvré", "XX-XX"), Spot()), 0.01)
+ SnowBall()
```

Calculer (3+4) * (4+2)



Quelques exemples

Calculer (3+4) * (4+2)

- En Java

```
Expression expression = new Multiply(new Add(new Const(3), new Const(4)),
                                     new Add(new Const(4), new Const(2)));

ExpressionVisiteur eval = new EvaluateVisiteur();
expression.evaluate(eval);
```

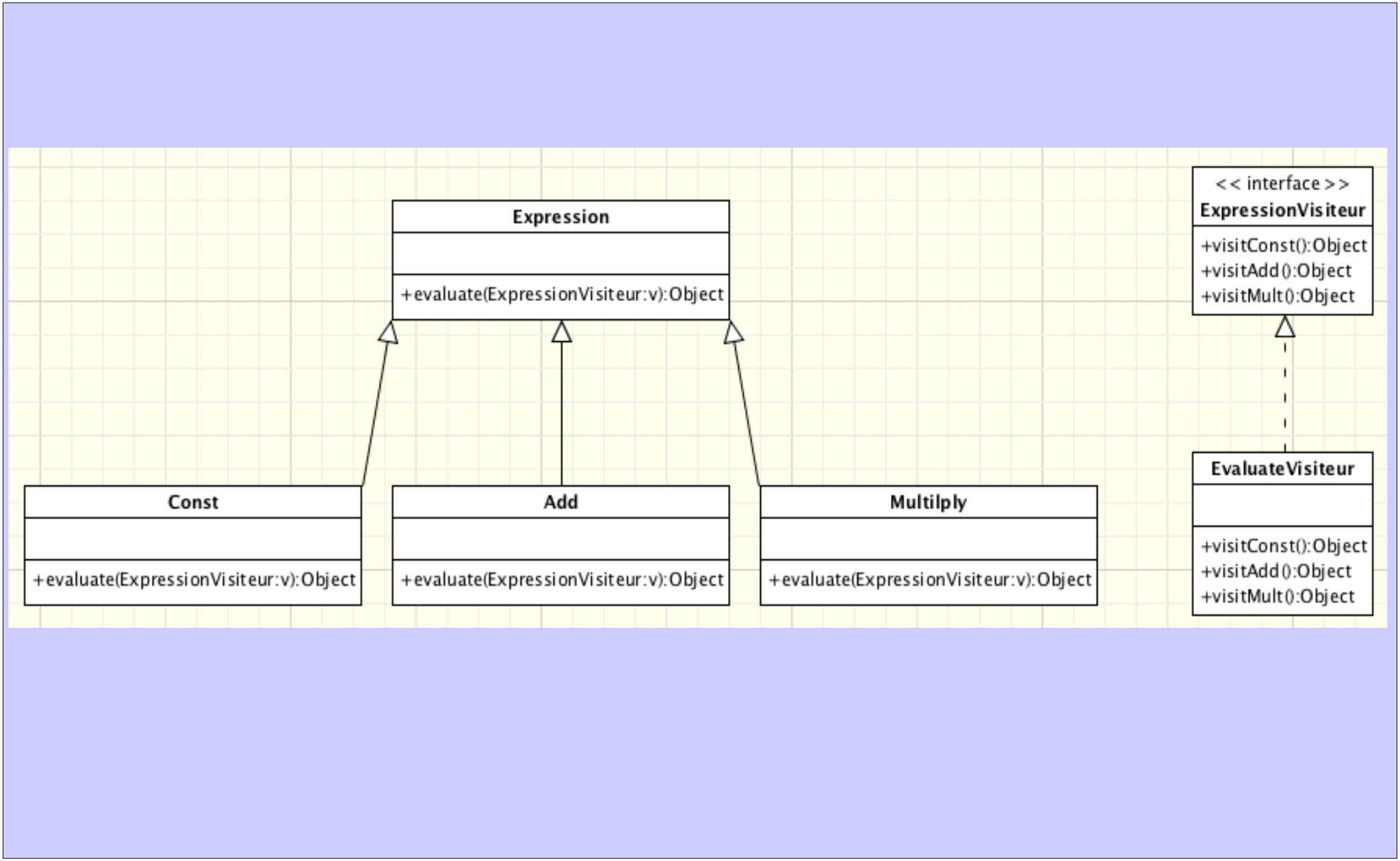
- En Scala

```
var expression =  Multiply(Add(Const(3), Const(4)),
                           Add(Const(4), Const(2)));

val v = new Visiteur();
v.evaluate(expression);
```

Pattern matching

```
abstract class Expr

case class Const(n: Int) extends Expr
case class Add(l: Expr, r: Expr) extends Expr
case class Multiply(l: Expr, r: Expr) extends Expr

class Visiteur {
  def evaluate(e: Expr): Int = e match {
    case Const(n) => n
    case Add(l, r) => evaluate(l) + evaluate(r)
    case Multiply(l, r) => evaluate(l) * evaluate(r)
 }
```

Pattern visiteur



Pattern visiteur

```
public abstract class Expression {
    abstract Object evaluate(ExpressionVisiteur v);
}


public class Add extends Expression {
    private Expression expr1;
    private Expression expr2;
    public Add(Expression expr1, Expression expr2) {
        this.expr1 = expr1;
        this.expr2 = expr2;
    }
    public Object evaluate(ExpressionVisiteur v) {
        return v.visitAdd(expr1, expr2);
    }
}
```

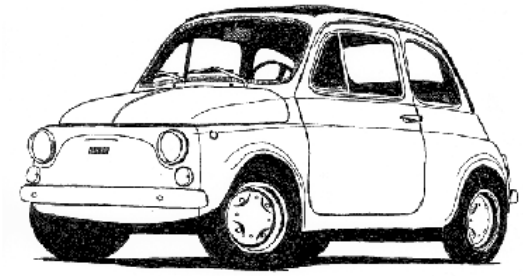
Pattern visiteur

```
public interface ExpressionVisiteur {
    Object visitConst(int c);
    Object visitAdd(Expression expr1, Expression expr2);
    Object visitMult(Expression expr1, Expression expr2);
}

public class EvaluateVisiteur implements ExpressionVisiteur {
    public Object visitConst(int constante) {
        return new Integer(constante);
    }

    public Object visitAdd(Expression expr1, Expression expr2) {
        return new Integer(((Integer) expr1.evaluate(this)).intValue()
                        + ((Integer) expr2.evaluate(this)).intValue());
    }
    ....
}
```

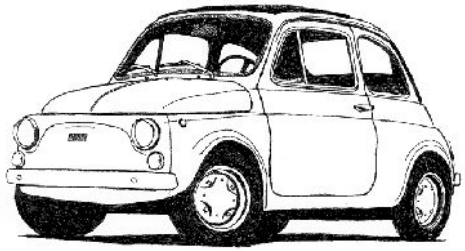
API Actors

- ***Un actor, c'est :***
  - **Un objet qui vit dans la JVM**
  - **Pile de messages type mailbox**
  - **Il reçoit des messages, et il répond…**
  - **Soit 1 thread par actor**
  - **Soit un pool de threads (plusieurs millions d'actors)**
  - **Remote actor (jvm distantes)**

Avant : threads



Après : actors



Les Actors

```
val printActor = actor {
   loop {
      react {
         case _ => println(text)
      }
   }
}

printActor ! "foo"
printActor ! "bar"
```

Les Actors

```
val pong = new Pong
val ping = new Ping(100000, pong)

ping.start
pong.start
```

Les Actors

```
class Ping(count: int, pong: Actor) extends Actor {
   def act() {
      pong ! Ping
      loop {
         react {
            case Pong => pong ! Ping
         }
      }
   }
}
```

Les Actors

```
class Pong extends Actor {
    def act() {
        loop {
            react {
                case Ping => sender ! Pong
            }
        }
    }
}
```

Les Traits

- ***Définition :*** **interfaces permettant de définir des méthodes/variables abstraites, mais aussi des méthodes concrètes**

Les Traits

```
trait XML {
  def toString(): String
  def toXML(): String = "<![CDATA[" + toString() + "]]>"
}

class MyObject {
  override def toString() = "Hello Paris JUG !"
}
```

```
var myObject = new MyObject with XML
println(myObject.toXML)

=> <![CDATA[Hello Paris JUG !]]>
```

Les Traits

```
abstract class Car {
  def drive() = {
    print("Drive an");
  }
}


trait OilCar extends Car {
  override def drive = {
     super.drive();
     println(" oil car");
  }
}


class MyCar extends OilCar {
}
```

```
trait ElectricalCar extends OilCar {
  override def drive = {
     super.drive();
     println(" electrical car");
  }
}


trait LogCar extends Car {
  abstract override def drive = {
    println("Entree peage");
    super.drive
    println("Sortie peage");
  }
}
```

Les Traits

```
var myCar = new MyCar
myCar.drive
=> Drive an oil car


myCar = new MyCar with ElectricalCar
myCar.drive
=> Drive an oil car
    electricial car


myCar = new MyCar with ElectricalCar with LogCar
myCar.drive

=> Entree peage
   Drive an oil car
    electricial car
   Sortie peage
```

Outillage

Plugin Eclipse

Refactoring quasi impossible

Coloration syntaxique, markers

Plante très souvent

Compatabilité java / scala, trait, fonction...

Difficulté au niveau configuration projet

Outillage

Plugin IntelliJ

Parfois lent à la compilation

Complétion très bien gérée

Très bonne gestion de l'inférence

Configuration de projet simple

Outillage

Plugin NetBeans

Installation laborieuse du plugin

Plugin en version Beta

Les frameworks

ScalaTest™
Less code, more clarity™

: tester en Scala en DSL !!!

```
class StackSpec extends FlatSpec with ShouldMatchers {
  "A Stack" should "pop values in last-in-first-out order" in {
    val stack = new Stack[Int]
    stack.push(1)
    stack.push(2)
    stack.pop() should equal (2)
    stack.pop() should equal (1)
  }

  it should "throw NoSuchElementException if an empty stack is
             popped" in {
    val emptyStack = new Stack[String]
    evaluating { emptyStack.pop() } should produce
                                    [NoSuchElementException]
  }
}
```

Les frameworks

AKKA : Actors++ !

- Tolérance aux pannes
- *Let it crash ! Don't try to prevent it, but manage it !*
- Stratégies de redémarrage
- Simplification des remote actors
- 2 à 3 fois plus rapides que les actors Scala
- Nombreux modules : Comet, REST, Security, Spring, Guice, AMQP, Scheduler...

Les frameworks

- Simple, concis, ajax, comet... focus sur le métier
- Tourne sur tomcat, jetty, weblogic...
- *Pas de code dans les pages HTML*
- Logique gérée dans le code Scala
- Snippet = tag librairies
- Mapping assez poussé (BD, model, snippet et pages HTML)

Pourquoi faut-il s'y intéresser ?

- Code plus concis
- Développement plus rapide (peut-être pas tout de suite)
- Tests encore plus métier
- Gestion « plus simple » des problèmes de modélisation
- Gestion de la concurrence

Difficultés/Challenges

- Enrichissement personnel

  => ça fait marcher le cerveau !

- Nouveau paradigme

- Penser fonctionnel, ça prend du temps

Les freins à son adoption

- Problème de maîtrise du langage
- Peut devenir assez vite complexe

```
def foo(l: List[int]): int = (0/:l){_+_}
```

- Courbe d'apprentissage
- Intégration avec Java
- Scala dans des parties business critical

Le futur

- Quelques nouveautés de Scala 2.8 :
  - Nouvelle API collection
  - REPL amélioré (complétion...)
  - Paramètres nommés et par défaut
  - Nouveaux contrôles (breakable and break)
  - Pas de compatibilité binaire avec 2.7

Qui utilise Scala ?

Conclusion

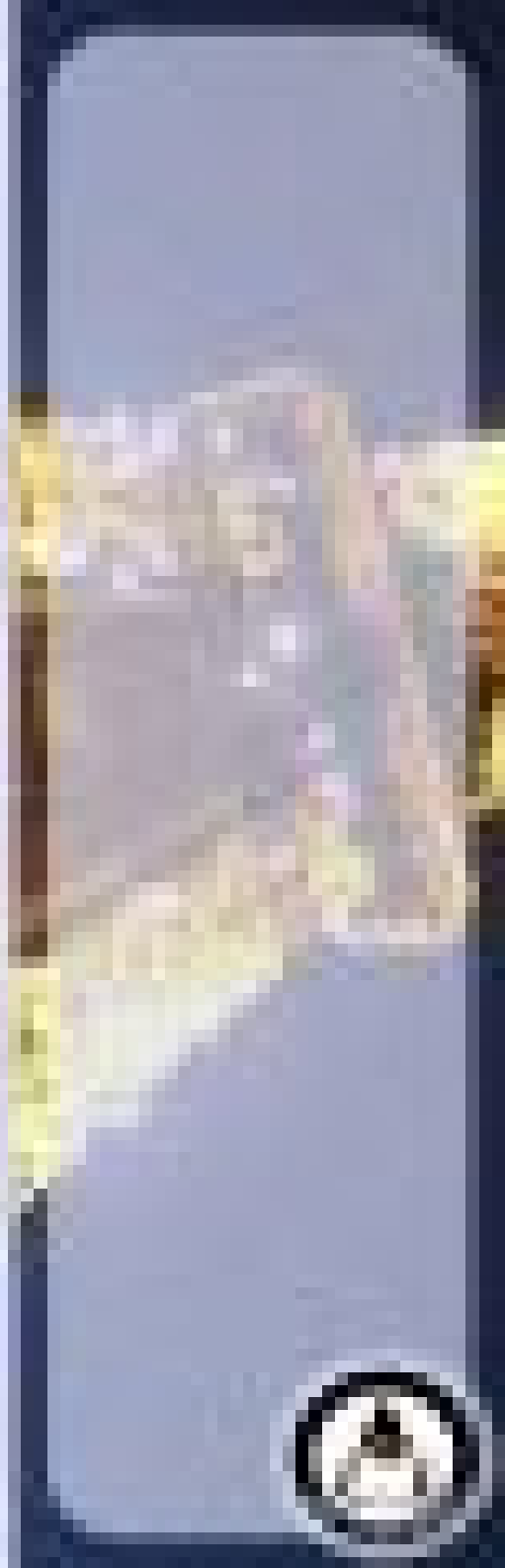
Essayez-le !!!

Bibliographie / liens

- http://www.scala-lang.org/
- http://www.artima.com/scalazine
- http://www.codecommit.com/blog/

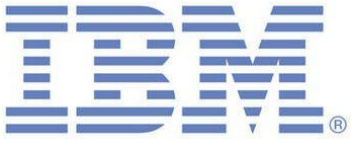
- Programming in Scala: A Comprehensive Step-by-step Guide (*Martin Odersky*)
- Programming Scala: Tackle Multicore Complexity on the JVM (*Venkat Subramaniam*)

Questions / Réponses



www.parisjug.org

Sponsors



Merci de votre attention!

www.parisjug.org

Licence

- http://creativecommons.org/licenses/by-nc-sa/2.0/fr/