Scala

Вадим Ипполитов

vadipp@gmail.com

twitter.com/vadipp

1.1. Обо мне

- Автоматизирую тестирование системы автоматизации хостинга в Parallels

- Участвую в создании хакспейса в Новосибирске
  - HackNsk.org

1.2. О Scala

- Язык общего назначения
- Объектно-ориентированный
- Функциональный
- Статически типизированный
- Расширяемый

- http://scala-lang.org/

1.3. О докладе

- Цель: ознакомление
  - Что есть в языке
  - Как оно выглядит
- Буду упрощать
- Результат: копать или не копать?

2.1. Статическая типизация

- Автоматическое выведение типов
  - Type inference

- Явно указывать типы значений приходится в тех местах, где их правда важно указать
  - Сигнатуры функций
  - Ещё кое-где :)

2.2. Пример

```
val greeting = "Hello world"
def shout(msg: String) =
        msg.toUpperCase

println(shout(greeting)) // HELLO WORLD
```

3.1. ООП + ФП

- Любое значение — объект некого типа
- Любая операция — вызов метода
  - 1 + 2 === 1.+(2)
  - list.add(elem) === list add elem
- Любая функция — значение
  - Можно хранить и передавать

3.2. ООП + ФП

```
def cube(x: Int) = x*x*x
val lengths = List(1, 2, 3)
val lengths = 1::2::3::Nil //то же

val volumes = lengths.map(cube)
// List(1, 8, 27)
```

4.1. ООП

- **`class`**
  - Конструктор и члены
  - Все члены имеют реализацию
- **`abstract class`**
  - Конструктор и члены, нельзя создать объект
- **`trait`**
  - Только методы и абстрактные члены

4.2. ООП

```
class Person(val name: String) {
  def speak(what: String) =
    println(s"$name says: $what")
}

new Person("Jim").speak("Hi") // Jim says: Hi
```

- Наследование, полиморфизм, инкапсуляция...

4.3. ООП — трейты

```
trait Speaker {
  def speak(what: String): Unit
}
trait Englishman extends Speaker {
  abstract override def speak(what: String) =
      super.speak(what + ", sir")
}

val jack = new Person("Jack") with Englishman
jack.speak("Hello") // Jack says: Hello, sir
```

4.4. ООП — цепочка трейтов

```
trait Pissed extends Speaker {
 abstract override def speak(what: String) =
  super.speak("AAARGH, " + what.toUpperCase + "!")
}


val jacob = new Person("Jacob") with Pissed
with Englishman
jacob.speak("enough")
// Jacob says: AAARGH, ENOUGH, SIR!
```

5.1. Pattern matching

```
trait Shape
case class Circle(r: Int) extends Shape
case class Rect(w: Int, h: Int) extends Shape
//---------------------------------------
def area(s: Shape): Double = s match {
  case Circle(r) => r*r*math.Pi
  case Rect(w, h) => w * h
  case _ => sys.error("Unsupported: " + s)
}
area(Circle(3))  //  28.27
area(Rect(2, 4))  //  8
```

5.2. Pattern matching

- Pattern matching можно определить самому
  - *Pimp my library*

- Для примера: класс List
  - Односвязный персистентный список

```
val empty = Nil
val list1 = 1::Nil
val list2 = 2::list1
```

5.3. Pattern matching — lists

```
def addPairs(xs: List[Int]) : List[Int] =
xs match {
  case x::y::tail => (x+y)::addPairs(tail)
  case _ => xs  //  один либо ноль элементов
}
//-------------------------
addPairs(List(1,2,3,4,5,6))
// List(3, 7, 11)
addPairs(List(1,2,3,4,5))
// List(3, 7, 5)
```

5.4. Pattern matching — regex

```
case class IssueID(key: String, number: Int)
object IssueID {
  def unapply(issueId: String): Option[IssueID] = {
    val regex = """([^-]+)-(\d+)""".r
    issueId match {
      case regex(k, n) => Some(IssueID(k, n.toInt))
      case _ => None
}}}
//----------------------------
val IssueID(issueId) = "SI-7889"
println(issueId)  //   IssueID(SI,7889)
```

6.1. Коллекции!

- Ништяки в стандартной библиотеке

```
def addPairs(xs: List[Int]) =
    xs.grouped(2)
        .map(_.sum)
          .toList
//--------------------------
addPairs(List(1,2,3,4,5,6))
// List(3, 7, 11)
addPairs(List(1,2,3,4,5))
// List(3, 7, 5)
```

6.2. Коллекции!

- Структуры данных
  - Изменяемые
  - Неизменяемые (персистентные)
- Много функциональных методов
- Расширяемые

7.1. Ещё в языке

XML-литералы

Хвостовая рекурсия

Каррирование

Вариантность

Структурные типы

Интерполяция строк (и даже больше)

Неявные параметры и преобразования

Макросы (добрые, типобезопасные)

Delimited continuations

7.2. Ещё в библиотеках

- Акторы (Akka)
- Веб-фреймворки
  - Play! (MVC)
  - Lift (view-centric)
- Доступ к БД
  - Slick (a-la LINQ.NET)
  - Squeryl (ORM)

Companies Using Scala

IBM

SONY
make.believe

twitter

LinkedIn

SIEMENS

xerox

Novell

outside.in

NASA

eDF

foursquare

amazon

UBS

TOMTOM

Office DEPOT
Taking Care of Business

guardian.co.uk

OPOWER

Bank of America

yammer

Thatcham

Autodesk

CREDIT SUISSE

8.2. Люди юзают

- Excelsior
- JarSoft
- "Центр Разработки Игр"

- Новосибирское Scala-сообщество:
  twitter.com/ScalaNsk | bit.ly/scalansk_forum
  - Встречи-лекции
  - Печеньки!
  - Записи предыдущих встреч на youtube.com/ScalaNsk

9.1. Почитать

- Programming in Scala, 2nd edition
  - Martin Odersky, Lex Spoon, Bill Venners