Scala v Kotlin v Java8

Jan Kotek

geecon

Prague, 20-21 October 2016

Jan Kotek

jan@kotek.net

@JanKotek

Tried to create alternative Scala Eclipse IDE

Author of MapDB (written in Kotlin)

Works at IOHK.io

What is missing

- N created in 2001?
- Loose syntax
- Has its own collection framework
- Its own eccosystem: Spark, Akka...

Java 8

- Created in 2013
- Backward compatibility with Java 7,6,5,4..
- Does not have some basic features
  - Type inference
  - Named arguments

Scala

- First release in 2003
- Created in academia
- Great features, highly expressive
- Independent on Java, but interoperaperable
  - Separate collection framework and other libraries
- Does not have native Loops and Loop Control Statements

Kotlin

- First released in 2012
  - stable 1.0 released in February 2016
- Created by JetBrains for ‘industrial use’
  - Included in recent Intellij Idea
- Inspired by Scala
- Great interoperability with Java

Method declaration

Scala

```
def abs( i:Int ) : Int = { Math.abs(i) }
def abs( i:Int ) = Math.abs(i)
```

Kotlin

```
fun abs( i:Int ) : Int { return Math.abs(i) }
fun abs( i:Int ) = Math.abs(i)
```

Value declaration

Scala

val str:String = "String"

var str = "String"

Kotlin

val str:String = "String"

var str = "String"

Lambda declaration

In editor

Data classes

Scala

case class Person(name: String, age: Int)

Kotlin

data class Person(val name: String, val age: Int)

Immutable, copy can alter parameters

Objects

- Objects are static singletons in Scala or Kotlin
- Scala objects can not be called directly from Java
- Kotlin needs *@JvmStatic*

In editor

Pattern matching

- Switch statement on steroids
- Java7+ can use Strings

In editor

Array style access

Scala

```
map("key")
map("key") = "value"
```

Kotlin

```
map["key"]
map["key"] = "value"
```

Extension functions

- Add new method to existing types
  - Only local
  - Activated by import
- Great for small Domain Specific Language

In editor

Weakly typed syntax

- Strongly typed languages
- Some problems in Java
  - 0123, 1L vs 1l
- Scala was designed to allow fluent syntax (as spoken language)
  - Dots are optionl when accessing members
  - Parenthesis are optional on methods and lambdas
  - Native XML support
- Kotlin is stricter
  - Bite-wise operators are renamed to methods
  - Lambdas always in {}

No loops in Scala

- Scala has no native loops
  - Implemented with lambdas
- No Loop Control Statements in Scala
  - break and continue

Conclusion

- Scala is highly expressive
  - Great for heavy lifting; compilers, AI, machine learning
  - DSL, Haskell style of programming…
- Kotlin is Scala without the annoying stuff
  - Great interoperability with Java
- Java8 allows functional programming
  - is compatible

Q&A

Jan@kotek.net

@JanKotek