Enjoy Learning Scala

vitojeng @ 亦思科技
2016.11.30

Agenda

- My book list
- Scala scripting
- Scala language
- Framework / Library / Tool
    - SBT
    - IntelliJ IDEA
    - Scalatra
    - ScalaTest

Start fighting

- Turning point
- Great imagination
- Like a chicken with its head cut off
- Keep going

Just in time

- Streaming / Concurrency & Parallel / BigData / Non-blocking
- Scala 2.8 +

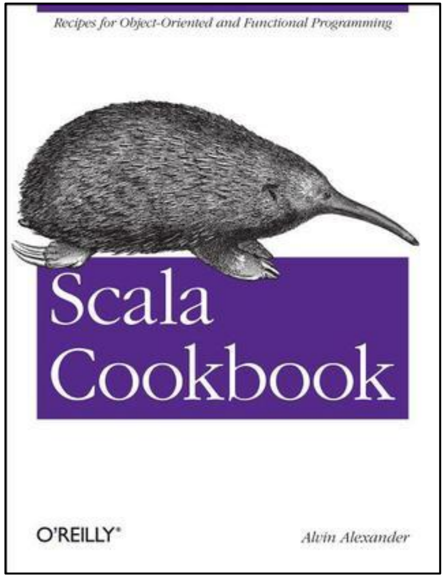
> ***JAXenter:*** *In your opinion, what are the most important technical milestones for this programming language?*
>
> ***Martin Odersky:*** *The most important step was no doubt Scala 2.8, which came out in 2010.*

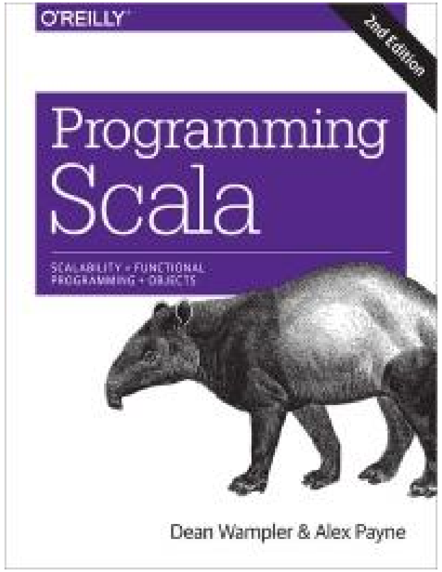
- Scala Taiwan
  - Meetup: *http://www.meetup.com/Scala-Taiwan-Meetup/*
  - *Chat room: https://gitter.im/ScalaTaiwan/ScalaTaiwan*

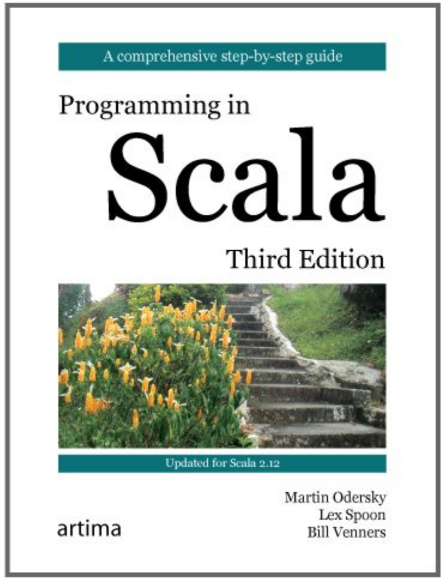
https://www.lightbend.com/company/news/jaxenter-interview-with-scala-creator-martin-odersky-on-the-current-state-of-scala

Scala book list

My scala book list




O'REILLY
2nd Edition
Programming Scala
SCALABILITY = FUNCTIONAL PROGRAMMING + OBJECTS
Dean Wampler & Alex Payne

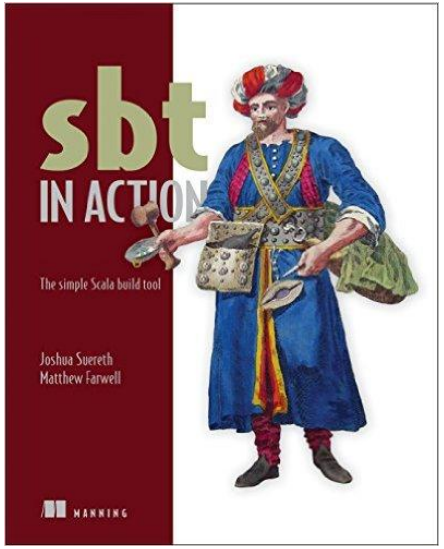


Scala Cookbook

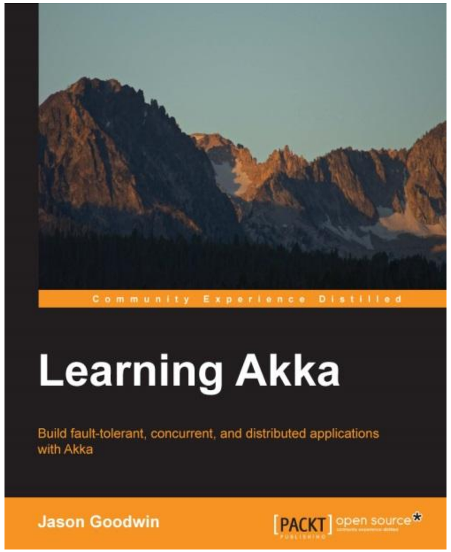
Programming Scala

Programming in Scala

My scala book list



sbt IN ACTION

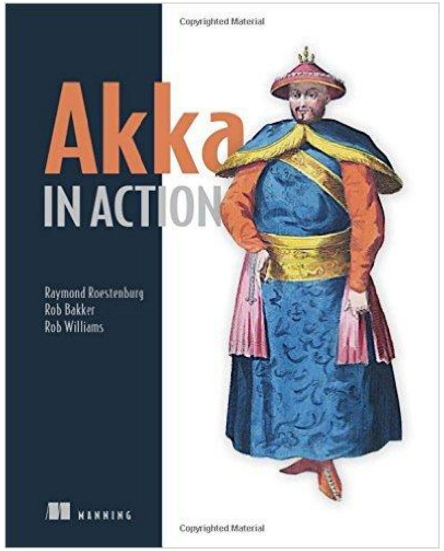

Community Experience Distilled
Learning Akka
Build fault-tolerant, concurrent, and distributed applications with Akka
Jason Goodwin
[PACKT] open source

Learning Akka

Community Experience Distilled
RESTful Web Services with Scala
Learn the art of creating scalable RESTful web services with Scala
Jos Dirksen
[PACKT] open source

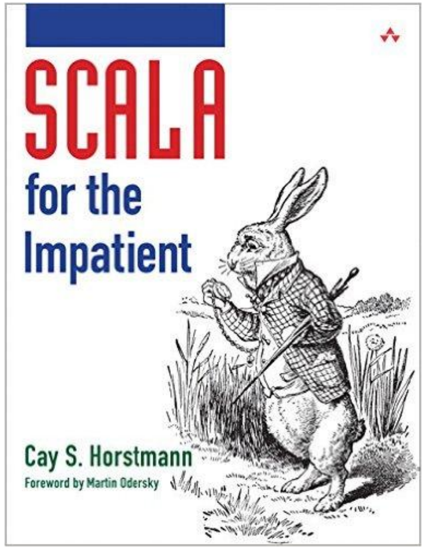
RESTful Web Services with Scala

My scala book list



Akka IN ACTION

Functional Programming in scala



SCALA for the Impatient

Scala Scripting

Scala REPL

- Run script

```
$ scala [scala-file]
```

- Self-executable

```
#!/usr/bin/env scala
```

- :load
- :paste
- :javap

SBT

- *sbt-launch.jar*
- script runner: *scalas*

```
java -Dsbt.main.class=sbt.ScriptMain
-Dsbt.boot.directory=/home/user/.sbt/boot -jar sbt-launch.jar "$@"
```

- Self-executable

```
#!/usr/bin/env scalas
```

- http://www.scala-sbt.org/0.13/docs/Scripts.html

Ammonite

- Ammonite lets you use the Scala language for scripting purposes: in the REPL, as scripts, as a library to use in existing projects, or as a standalone systems shell.
- Syntax Highlignting / Pretty-printed output / Multi-line editing / ...
- Install

```
$ sudo curl -L -o /usr/local/bin/amm https://git.io/vXVf5
$ sudo chmod +x /usr/local/bin/amm && amm
```

- Video - https://vimeo.com/191328477

Scripting - Ammonite

- Magic imports: ***$file***, ***$ivy***
- Running from REPL
- Running from bash
  - command: `$ amm scripting/github2.scala`
  - self-executable: `#!/usr/bin/env amm`
- If meet Ivy resolution exception
  - *ammonite.runtime.tools.IvyThing$IvyResolutionException*
  - try to remove files in ***~/.ivy2/cache***

Scala Language

String

- String equality

```
val s1 = "123"
val s2 = "123"
val s3 = new String("123")
s1 == s2
s2 == s3
s1 eq s2
s2 eq s3
```

- Multi-line

```
test("SPARK-7319 showString") {
 val expectedAnswer = """+---+-----+
                        ||key|value|
                        |+---+-----+
                        ||  1|    1|
                        |+---+-----+
                        |only showing top 1 row
                        |""".stripMargin
 assert(testData.select($"*").showString(1) ===
expectedAnswer)
}
```

String Interpolation

- s / f / raw

```
val language = "scala"
val dd = new Date()
println( s"Hello, $language" )
println( s"Hello, ${language.toUpperCase} " )
println( f"Hello, $language%s. Time: $dd%tY/$dd%tm/$dd%td" )
println( "first line\nsecond line" )
println( raw"first line\nsecond line" )
```

- Custom

```
import better.files._
import java.io.{File => JFile}
val f = File("/User/johndoe/Documents")
val f1: File = file"/User/johndoe/Documents"
```

https://github.com/pathikrit/better-files

Type inference

- Pros: Large reduce the code size.
- Cons: Sometimes may reduce the code readability.
- Balance pros and cons.
- Using Intellij-IDEA to view code.

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

val (i, f, s) = (100, 99.0, "Hello")
val list = List(1, 2, 3, 4, 5)
val people = Seq(Person("john", 40), Person("jack", 28), Person("ann", 24))

val youngPeople = people.filter { case Person(n,a) => a <= 30 }
```

Implicits

- Add own methods to exist object

```scala
implicit class DateConvert(val date: Date) {
 private def cloneDate(date: Date, f: Calendar => Unit) = {
   val cal = Calendar.getInstance()
   cal.setTime(date)
   f(cal)
   cal.getTime
 }
 def firstDayOfMonth() = cloneDate(date, _.set(Calendar. DAY_OF_MONTH, 1) )
 def firstDayOfWeek() = cloneDate(date, _.set(Calendar. DAY_OF_WEEK, 1) )
}
val mydate = new Date()
println( mydate )
println( mydate.firstDayOfMonth() )
println( mydate.firstDayOfWeek() )
```

Mutable / immutable objects

- Scala made immutable class more easy !

```
case class Address(street: String="", city: String="",
                   state: String="", zip: String="")

case class Person(name: String, age: Int = 0,
                 sex: Char = 'M', address: Address = Address())

val p1 = new Person(name="joseph")
val p2 = new Person(name="anna", sex='F')
```

Mutable / immutable objects

Pessimistic copying can become a problem in large programs. When mutable data is passed through a chain of loosely coupled components, each component has to make its own copy of the data because other components might modify it. ***Immutable data is always safe to share, so we never have to make copies***. We find that in the large, FP can often achieve greater efficiency than approaches that rely on side effects, due to much greater sharing of data and computation.

- *Functional Programming in Scala, chapter 3*

- If immutable objects are good, why do people keep creating mutable objects?

Mutable / Immutable Collections

- More mutable/immutable support:
  - scala.collection.mutable
  - scala.collection.immutable
- Scala documation: Mutable and Immutable Collections
- Book: 深入探索Scala集合技術手冊 (松崗)

Pattern matching

- Pattern matching very powerful, must try it !!

```
val manyObjects: Seq[Any] = Seq("scala", "2.11.8", 18, 120, 1.5)

manyObjects.foreach { x =>
 val message = x match {
   case i: Int if i < 100 => "(int)(less than 100) " + i
   case j: Int => "(int) " + j
   case "scala" => "scala !!"
   case s: String => "(string) " + s
   case _ => "(not handle) " + x
 }
 println(message)
}
```

Pattern matching

```
case class Address(street: String, city: String, country: String)
case class Person(name: String, age: Int, address: Address)
val alice   = Person("Alice",   25, Address("1 Scala Lane", "Chicago", "USA"))
val bob     = Person("Bob",     29, Address("2 Java Ave.",  "Miami",   "USA"))
val charlie = Person("Charlie", 32, Address("3 Python Ct.", "Boston",  "USA"))

for (person <- Seq(alice, bob, charlie)) {
 person match {
   case Person("Alice", 25, Address(_, "Chicago", _)) => println("Hi Alice!")
   case Person("Bob", 29, Address("2 Java Ave.", "Miami", "USA")) =>
     println("Hi Bob!")
   case Person(name, age, _) =>
     println(s"Who are you, $age year-old person named $name?")
 }
}
```

https://github.com/deanwampler/prog-scala-2nd-ed-code-examples/blob/master/src/main/scala/progscala2/patternmatching/match-deep.sc

Functioal Programming(term)

- Function's side effect
  - Modify a variable
  - Modify a data structure in place
  - Setting a field on an object
  - Throwing an exception or halting with an error
  - Printing to the console or reading user input
  - Reading from or writing to a file
  - Drawing on the screen
- Pure function - Functions that have no side-effects

- *Functional Programming in Scala, chapter 1*

Functioal Programming(term)

- High order functions(HOFs)
  - passing function to functions
  - functions are values
- Currying
- Function composition
  - feeds the output of one function to the input of another function.

More features

- Trait & Compound Type

```
trait T1
trait T2
class C
val c = new C with T1 with T2
```

- DSL: an XML example
- Java integration support
- Concurrency

Framework / Library / Tool

SBT - The interactive build tool

- The interactive build tool.
- More complex than Gradle & document not good.
- Fast compilation.
- Cross-compilation, across several scala versions.
- Continue compilation/testing: **~**
- Test one class: ***~testOnly *YourClass***
- Test one method: ***~testOnly * YourClass -- -z "method name"***
- Integrate Scala REPL: ***console***
- Gradle or SBT ?
    - If you want stay in Gradle, *Kafka* maybe a good reference.

Intellij IDEA - IDE

- The best Java IDE I ever used.
- Enable Scala support: install Scala plugin
  - Scala Worksheet
  - SBT, PlayFramework
  - SSP(Scala Server Pages)
  - HOCON(Typesafe's configuration format)
  - ScalaTest, spacs2
- https://www.jetbrains.com/help/idea/2016.3/scala.html
- Import ***project*** or ***module*** from SBT

Scalatra - Web framework

- A port of the Sinatra framework written in Ruby.
- As a Scala beginner, I choice *Scalatra* rather than *Spray(Akka-http)* or *PlayFramework* in my job.
- Hightlights
  - Base on Java Servlet technology
  - Integrate with SBT
  - View: Inline HTML, Scalate, Twirl
  - Async: AkkaSupport
  - Persistence: no built-in integrations.
  - JSON: json4s
  - Test: ScalaTest, Specs2
  - Deployment: Standalone, Servlet Container

Scalatra - Web framework

- Version information
  - Scalatra 2.3.0(2014-06): support Scala 2.10, Servlet 3.1
  - Scalatra 2.4.0(2015-12): support Scala 2.10, 2.11
  - Scalatra 2.5.0(2016-11): support Scala 2.12
- Example

ScalaTest

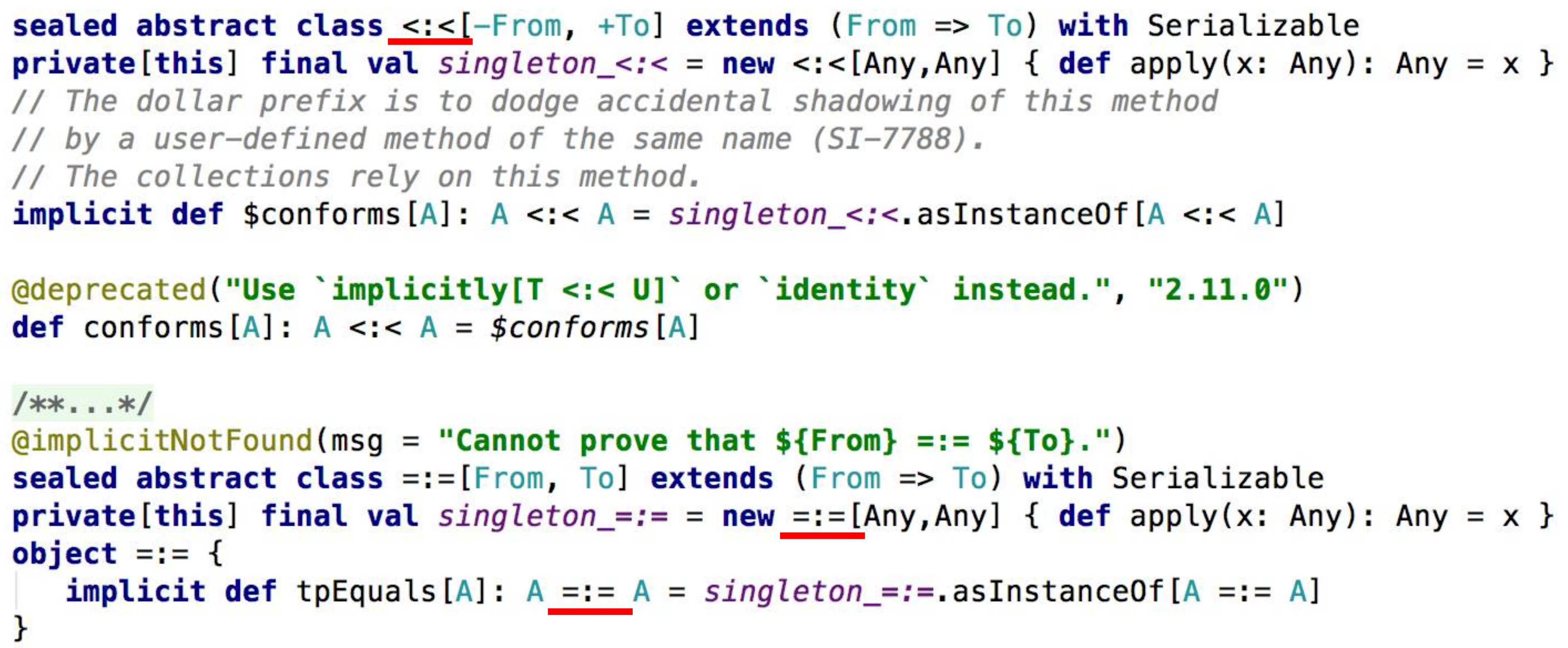
- Many project using ScalaTest: *Apache Spark*, *Akka* ...
- For new test case: consider not using JUnit
- For exist JUnit test case: use SBT with junit-interface
- Still want using JUnit in Scala ?
  - Try *org.scalatest.junit.JUnitSuite*
  - *Apache Kafka* is good reference using *JUnitSuite*
- BeforeAndAfterAll

Scala.js

- https://www.scala-js.org/

Funny things (Terrible ?)

Text-face programming ?



```scala
sealed abstract class <:<[-From, +To] extends (From => To) with Serializable
private[this] final val singleton_<:< = new <:<[Any,Any] { def apply(x: Any): Any = x }
// The dollar prefix is to dodge accidental shadowing of this method
// by a user-defined method of the same name (SI-7788).
// The collections rely on this method.
implicit def $conforms[A]: A <:< A = singleton_<:<.asInstanceOf[A <:< A]

@deprecated("Use `implicitly[T <:< U]` or `identity` instead.", "2.11.0")
def conforms[A]: A <:< A = $conforms[A]

/**...*/
@implicitNotFound(msg = "Cannot prove that ${From} =:= ${To}.")
sealed abstract class =:=[From, To] extends (From => To) with Serializable
private[this] final val singleton_=:= = new =:=[Any,Any] { def apply(x: Any): Any = x }
object =:= {
    implicit def tpEquals[A]: A =:= A = singleton_=:=.asInstanceOf[A =:= A]
}
```

Text flow-diagram programming ?

```
A ~> B ~> C ~> D
          E <~ D
     G <~ F <~ D
```

https://gist.github.com/rocketpages/9ee1523db698edc4cfb5#file-flightdelays-scala

Toy language implementation ?

```
object SquareRoot extends Baysick {
   def main(args:Array[String]) = {
     10 PRINT "Enter a number"
     20 INPUT 'n
     30 PRINT "Square root of " % "'n is " % SQRT('n)
     40 END

     RUN
   }
 }
```

https://github.com/fogus/baysick/tree/master

Thank you !

Don't wait for the future. Invent it.

Scala Taiwan

Meetup: *http://www.meetup.com/Scala-Taiwan-Meetup/*
Chat room: *https://gitter.im/ScalaTaiwan/ScalaTaiwan*