JDevDay - Introduction to Scala

Saleem Ansari

10th March 2013

Outline

1. Introduction to Scala
2. Setting up Scala
3. Absolute Scala basics
4. Functional Paradigm in Scala
5. Object Oriented Programming in Scala
6. Where to learn more Scala

Topic

1. Introduction to Scala
2. Setting up Scala
3. Absolute Scala basics
4. Functional Paradigm in Scala
5. Object Oriented Programming in Scala
6. Where to learn more Scala

What is Scala?

What is Scala? The name Scala stands for "scalable language". The language is so named because it was designed to grow with the demands of its users.

- What makes Scala a Scalable Language ?
  - Statically typed language with OOP + FP both.
  - Fast and expressive, and peasant to use.
  - Excellent type inference.
  - Runs on JVM.
- Whats more?
  - Scala is compatible with Java
  - Scala is concise
  - Scala is high level
  - Scala is statically typed (vs dynamically typed)

What can you do with Scala today?

- Write web applications
  - Play Web Framework http://www.playframework.com/
  - Lift Web Framework http://liftweb.net/
- Write code that scales to huge amounts of data
  - Spark Project http://spark-project.org/
  - Scalding https://github.com/twitter/scalding
- Process huge number of concurrent tasks
  - Akka http://akka.io/
- Natural Language Processing and Machine Learning
  - ScalaNLP http://www.scalanlp.org/
- And anything could do in Java, now more concisely :-)

Topic

1. Introduction to Scala
2. Setting up Scala
3. Absolute Scala basics
4. Functional Paradigm in Scala
5. Object Oriented Programming in Scala
6. Where to learn more Scala

Install the Scala Complier

- yum install scala

Install your faviourite Editor / IDE

- yum install emacs
- OR install Eclipse (ScalaIDE)

Scala compiler is

- scalac (just like javac command)
- scala (just like java command)
- fsc (fast scala compiler)

Topic

1. Introduction to Scala
2. Setting up Scala
3. Absolute Scala basics
4. Functional Paradigm in Scala
5. Object Oriented Programming in Scala
6. Where to learn more Scala

Hello Scala World

helloworld.scala

```
object HelloWorld {
  def main(args: Array[String]) = {
    println("Hello Scala World!")
  }
}
```

Compile and run Hello Scala World

Output

```
$ scalac helloworld.scala
$ ls
helloworld.scala
HelloWorld.class
HelloWorld$.class
$ scala HelloWorld
Hello Scala World!
```

Values and Variables

- An example in Ruby (or maybe Python) a dynamically typed language
  - counter = Counter.new
  - counter = AtomicCounter.new
  - counter = File.new # this works here!
- Scala's static type system, avoids runtime overhead of dynamic types. The method dispatch is fast in a statically typed language.
  - var counter = new Counter()
  - counter = new AtomicCounter() // this has to be a Counter
  - counter = new File() // this doesn't work in Scala

Data Types

- Almost everything is same as Java
- Basic Data Types: (all integers are signed two's complement)
  - Integers: **Byte** (8bit), **Short** (16bit), **Int** (32bit), **Long** (64bit)
  - **Char** (16 bit unicode character), **String** (squence of Chars)
  - Reals: **Float** (32bit), **Double** (64bit)
  - **Boolean**: true / false
- Literal
  - basic data types i.e. 1, 0.123, 12L, 'a', "String"
  - symbol literal: 'identifier

More Concepts

- Data Containers
  - Array
  - List
  - Set
  - Map
  - Tuple
- Programming Abstraction Tools
  - Class
  - Object
  - Scala App
  - Package

Expressions

- Every thing is an expression
  - Basic expression: 1+2
  - An assignment is an expression
  - A function is an expression

Control Constructs

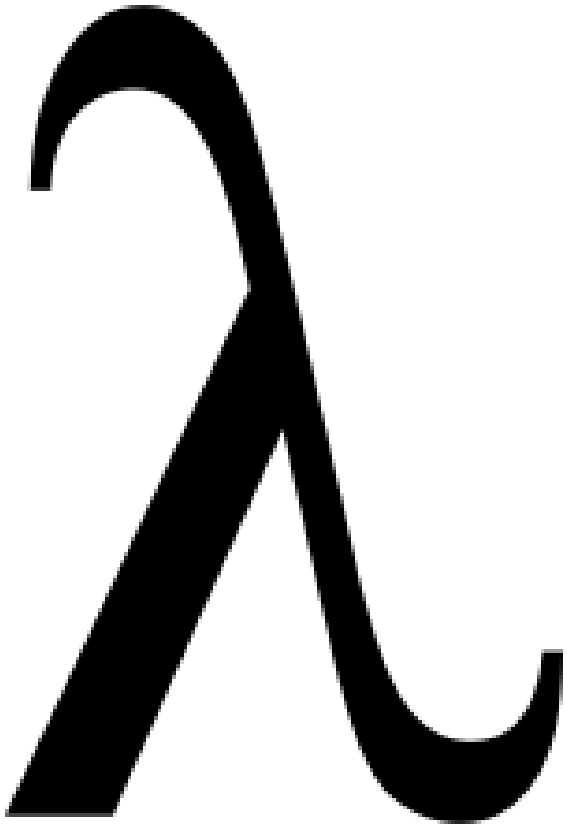
- if-else
- while
- do-while
- for
- match-case
- try-catch-finally

Topic

1. Introduction to Scala
2. Setting up Scala
3. Absolute Scala basics
4. Functional Paradigm in Scala
5. Object Oriented Programming in Scala
6. Where to learn more Scala

Matematical Logic

- Lambda Calculus (see Wikipedia)



Factorial Function

- Expressed as mathematical logic

$$n! = \begin{cases} 1 & \text{if } n = 0, \\ (n-1)! \times n & \text{if } n > 0. \end{cases}$$

FP is guided by two main ideas:

- Functions are **fist-class values**
- Functions have **no side effects** i.e. they can be replaced with their values

Hallmarks of Functional Programming

- mapping
- filtering
- folding
- reducing

Topic

1. Introduction to Scala
2. Setting up Scala
3. Absolute Scala basics
4. Functional Paradigm in Scala
5. Object Oriented Programming in Scala
6. Where to learn more Scala

Object Oriented

- Decompose the problem into entities and interactions among entities
- Each entity and their interaction is represented using class/object
  - internal state is the member variables
  - interactions are the member functions

Functions

factorial.scala

```
def factorial(n:Int): Int =
  if(n<=0) 1 else n*factorial(n-1)
```

- Placeholder syntax
- Partially applied functions
- Closures

Traits

traits.scala

```
trait PartTime {
  // trait definition
}
```

Classes

classes.scala

```
class Employee(name: String, age: Int) {
  override def toString = name + ", " + age
}
class Supervisor(name: String, age: Int
  ) extends Employee(name, age) with PartTime
  {
  override def toString = name + ", " + age
}
```

Objects

objects.scala

```
object Employee {
  override def toString = name + ", " + age
}
```

Packages

package-example.scala

```
package in.tuxdna.scala
class Employee(name: String, age: Int) {
  override def toString = name + ", " + age
}

object Main extends App {
  val emp1 = new Employee("Tom", 21)
  println("Employee 1: "+emp1)
}
// $ scalac pacakge-example.scala
// Employee 1: Tom, 21
```

Features to be convered later

- XML Processing
- Actors
- Case Classes
- Properties
- Extistential Types
- Implicits
- Lazy Evaluation
- Parser Combinations
- Monads
- Annotations

Using Scala as a scripting language

- scala scriptname.scala

employee.scala

```
class Employee(name: String, age: Int) {
  override def toString = name + ", " + age
}

val emp1 = new Employee("Tom", 21)
println("Employee 1: "+emp1)

// $ scala employee.scala
// Employee 1: Tom, 21
```

Topic

1. Introduction to Scala
2. Setting up Scala
3. Absolute Scala basics
4. Functional Paradigm in Scala
5. Object Oriented Programming in Scala
6. Where to learn more Scala

Books

- **Scala for the Impatient** (free) http://blog.typesafe.com/free-pdf-from-typesafe-scala-for-the-impatien-64715
- **Programming Scala** (free) http://ofps.oreilly.com/titles/9780596155957
- **Programming in Scala 2nd Ed.** http://www.amazon.com/Programming-Scala-Comprehensive-Step-Step/dp/0981531644
- **Functional Programming Principles in Scala** (free online course)
  - https://www.coursera.org/course/progfun
- Blogs
- Forums

Questions?

- tuxdna (at) gmail (dot) com

Thank you!

- twitter.com/tuxdna