Speaker Notes: Scala for Apache Spark

Markus Dale, medale@asymmetrik.com

Jan 2019

Setup

- Open Spark API:
  https://spark.apache.org/docs/latest/api/scala/index.html
- Open Scala API: https://www.scala-lang.org/api/2.11.8/#package
- Open Java API String: https://docs.oracle.com/javase/8/docs/api/

Intro, Slides And Code

- Bio:
  - mostly Java, big data with Hadoop
  - big data with Spark, Databricks, Scala
  - Now Asymmetrik - Scala, Spark, Elasticsearch, Akka...
  - Data Engineer
- Slides: https://github.com/medale/scala-spark/blob/master/presentation/ScalaSpark.pdf
- Scala Spark Code Examples: https://github.com/medale/scala-spark
- Also https://github.com/medale/spark-mail

Goals

- Intro to Scala (from Java) to leverage Apache Spark with Scala API
- sbt - build tool
- spark-testing framework for integration testing

Why Scala for Spark?

- Data Engineer - scalable ecosystem of Java/Scala-based tools
- less boilerplate/less typing (Ted Malaska (three big data books on O'Reilly): 50% less than Java)
- strong typing, elegant multi-paradigm language (functional and OO)
- all code runs in executor JVM - no callouts to local Python shell for UDFs/UDAFs
- spark-shell is Scala-based
- Baltimore Scala meetup

Java to Scala - Java Main

- semicolons
- get/set JavaBeans convention
- explicit constructor
- static main method

Scala Main One

- Look Ma - no semicolons
- package structure - match directory structure
- Match file name/class name
- object vs. class
    - object - Java static methods, singleton
- no public (default)
- def - method/function declaration
- type declared after variable name
- parameterized type: `Array[String]`
- return type, body (last entry gets returned)
- val - immutable, var - mutable
- class constructor (args none vs. val vs. var)
    - Java get/set: `import scala.reflect.BeanProperty`
    - `@BeanProperty var firstName`

Scala Main One - Output

- println statements
- default class to String - fully qualified class name@...

Scala Main Two - case class

- immutable data structure
- default constructor parameters are `val`
- generates boiler plate code, singleton object
    - apply method - constructor without new
    - also unapply for pattern matching
- Scala: Lots of “syntactic sugar” - less typing, compiler translates

Scala Main One - Output

- toString method created, prints class name, instance variables

Scala Main Two - javap ScalaMainTwo.class

- javap disassembler (package names removed)
- implements Product (abstract algebraic type), Serializable
  - Serializable important for Spark shuffle!
  - Product
    - productArity
    - productElement(int)
    - productIterator
- static apply factory method/unapply for matching
  - two fields etc.: `public static Option<Tuple2<Object, Object>> unapply(Foo)`
- copy method
- equals, hashCode, toString (see javap output next)

HelloSparkWorld - expression-oriented

- expressions returns value (vs. statements)
  - Array (any indexed sequence) accessor args(index)
  - type of lines is inferred as lowest-common denominator if/else block
  - In this case: `Seq[String]`

Scala Type Hierarchy

- Main division - AnyVal vs. AnyRef (unified through Any)
- AnyRef is like Object in Java
  - Null is a subclass of all reference classes
- AnyVal - Java primitives
  - Unit - `val u = () //u: Unit = () 0-tuple`
  - Value class: `class Wrapper(val underlying: Int) extends AnyVal`
- Universal trait:
  - trait that extends Any
  - only has defs as members
- Nothing is subclass of everything (throwing exception returns Nothing)

HelloSparkWorld - SparkSession

- object - main method entry point
- SparkSession Scala API
    - scaladocs
    - object
    - class
    - .builder method (don't need empty parentheses - mutator method with ())
- Builder class - fluent interface/method chaining
    - getOrCreate
    - Run from shell or batch spark-submit

SparkSession Scala API

- Showed SparkSession scaladocs

HelloSparkWorld - String, StringOps, implicits

- Triple quoted string can include special chars like newline, double-quote
- .stripMargin - by default uses pipe | removes all chars in front of pipe
- String - where does stripMargin method come from? (next slide)

Java API - String

- java.lang.String - does not have stripMargin method

Scala Predef API - implicit conversions

- `implicit def augmentString(x: String): StringOps`
- Also: `implicit def booleanArrayOps(xs: Array[Boolean]): ArrayOps[Boolean]`

Scala StringOps API - stripMargin

- stripMargin method
- Also useful: head, tail, map, filter, sliding (ngrams), permutations

HelloSparkWorld - accessing Java API/libraries

- import aliasing: `import java.util.{List => JavaList}`
- `readLinesFromFile` uses nio.Paths/Files (could use any 3rd party Java library, e.g. Apache Commons IO)
- `import scala.collection.JavaConverters._` - implicit conversions
  - underscore like Java * - import all methods from JavaConverters
  - DecorateAsScala/DecorateAsJava/AsScala/AsScala
- last line `lines` gets returned from method (return type Seq[String])

wordCountLocal: map higher-order function w/named function

- Scala 1 JVM processing - Seq trait (Array implements)
- higher-order functions - input is a function (or returns function)
- map - iterate over seq, one input, one output element
    - Immutability: underlying seq is not updated, returns new seq!
- map - with named function
    - if small function - pollute namespace, harder to read

wordCountLocal: map higher-order function w/ function literal

- syntactic sugar 1 - type is inferred
- placeholder syntax - if parameter is only used once
    - for two param function, first , *second*

wordCountLocal: flatMap and filter * flatMap - one input element - GenTraversableOnce (Seq-like), 0 or more * filter - keep elements that test true

Scala Seq trait API

- Show filter, flatMap higher-order functions

wordCountLocal: groupBy, mkString

- groupBy - create map with word, list of all occurrences of word (could group by same starting letter...)
- mapValues(syntactic sugar - _ argument one)
- mkString
- string interpolation - s“...”

HelloSparkWorld - RDD map, flatMap, filter

- process at scale!
- parallelize from driver to executor
- Immutable RDD - map, flatMap, filter transformations!
    - action like collect, take, write causes execution

HelloSparkWorld - RDD of tuples - PairRDDFunctions

- groupBy - expensive shuffle operation
- map to 2 tuple - implicit conversion to PairRDDFunctions
  - object RDD - `implicit def rddToPairRDDFunctions(rdd: RDD[(K, V)])`
- PairRDDFunctions - reduceByKey - function with two arguments
  - local combine step, then shuffle (hashPartitioner)
- transformations (lazy) executed by action collect()!
  - to local driver - memory!!!

Want to cover - highlights but have in-depth examples in repo

- Intellij Scala plugin
- object/main method
- case class - Product
- functions - defining a function, anonymous functions
- collections - map, flatMap, filter
- immutability
- implicits - Predef / StringOps / StringLike
- Scala docs
- Spark - RDD, Dataframe, Dataset (Tungsten memory, code gen)
- SparkSession, DataframeReader
- udf
- sbt build - quick overview
- integration testing - spark-testing-base