Recap

Saturday, August 6, 2022 10:07 PM

1. Tuples
2. Sets
3. Maps
4. Auxiliary Constructor
5. Single Ton Objects
6. Companion Objects

Topics for Today

Sunday, July 3, 2022 7:52 PM

1. Case Class
2. Pattern Matching
3. Inheritance
4. Traits
5. Layered Traits
6. Higher Order Functions
7. Spark Components
8. Spark Architecture
9. Deployment Mode

Case Class

05 July 2020 14:23

Defination

Used for comparison of objects
Instances of case classes are compared by structure and not by reference:

Example

```
case class learnCaseClass(isbn:String)

val frank = learnCaseClass("987-123")
```

How it is different from a class

```
class learnCaseClass1(isbn:String)

val frank = new learnCaseClass1("987-123")
```

Pattern Matching

05 July 2020 15:01

Used for matching values of Variables

Definition

Keyword
match
case

Example 1

```
def matchTest(x:Any) : Any = x match {
   case 1 => "one"
   case "two" => 2
   case y: Int => "scala.Int"
   case _ => "many"
      }

println(matchTest("two"))
println(matchTest("test"))
println(matchTest(1))
```

Example 2

```
case class Person(name: String, age: Int)
   val alice = Person("Alice", 25)
   val bob = Person("Bob", 32)
```

```
  val charlie = Person("Charlie", 32)
  for (person <- List(alice, bob, charlie)) {
    person match {
      case Person("Alice", 25) => println("Hi Alice!")
      case Person("Bob", 32) => println("Hi Bob!")
      case Person(name, age) => println("Age: " + age
+ "year, name: " + name + "?")
    }
  }
```

Inheritance

05 July 2020 15:48

What is Inheritance?

Class A
    Var 1
        Var 2
    Function 1
    Function 2
Class c
Var

Var
Func
func

Class B extends Class C
   var 3
   Function 3

Keyword ?

Example

```
class learnInheritance (speed:Int) {
  val mph: Int = speed
  def race() { println("Racing")}
println("This is vehicle")
}
```

Keyword Extends

```
class Car(speed:Int)  extends
learnInheritance(speed)
```

Requirement : to change some of the features not all

Keyword : Override - To modify existing property of method

```
class Car(speed:Int)  extends
learnInheritance(speed) {
```

```
  override val mph: Int = speed
  override def race() = println("Racing Car")

def carMethod () { println("I am in class Car")

}

val a = new Car(20)
println("Speed of Car: " + a.mph)
a.race()
```

Traits

05 July 2020 15:55

Definition
To Reuse Code

Example of Trait

```
trait learnTrait {

//abstract functions
//Name /Signature is given
//Implementation not given
  def hasNext:Boolean
  def next():Int
}

class IntIterator (to: Int) extends learnTrait {
  private var current = 0
  def hasNext:Boolean = current < to
 def next(): Int = {
    if (hasNext) {
      val t = current
      current += 1
```

```
      t
    } else 0 }

}
val iterator = new IntIterator(10)
iterator.next()
iterator.next()
```

layered Traits

05 July 2020 18:05

Definition

Multiple traits can be extended by a class or object

with - Keyword

super - keyword

Example of layered Traits

```
//trait 1
trait logger  {
//a simple method log
  def log (msg: String) {println(msg)}
}

//trait 2
trait TimestampLogger extends logger {
  override def log (msg: String) {
println("We are in Timestamp Logger")
    println(new java.util.Date() )
```

```
    super.log(new java.util.Date() + " " + msg)
  }
}

//trait 3
trait ShortLogger extends logger {
  val maxLength = 15
   override def log(msg: String) {
println("We are in Short Logger")
    super.log( if (msg.length <= maxLength) msg
    else msg.substring(0,maxLength -3) + "...")
  }
}
//Notice the keyword extends , with
class logging extends  TimestampLogger with ShortLogger
val a = new logging
a.log("My example")
```

Higher Order Functions

08 July 2020 22:18

Functions that take other functions as parameters or return a function as a result

Example
Problem Statement: Input 2 int
Output sum of cube of 2 integers
(1,2)
1+8 = 9

Functions That accepts Function

Higher Order Function

```
def sum(f:Int => Int,a:Int, b: Int) : Int =
   if (a > b) 0 else f(a) + sum(f,a+1,b)


def cube(x:Int): Int = x * x * x

def sumCubesHO(a:Int, b: Int) = sum(cube,a,b)
```

```
def square(x:Int) : Int = x *x
def semSquaresHO(a:Int,b:Int) = sum(square,a,b)
```

```
def fact(x: Int) : Int = if (x==0) 1 else x*fact(x-1)
def sumFactorialHO(a:Int, b: Int) = sum(fact,a,b)
def sumofnum(x :Int , y :Int, f:Int => Int) : Int = {if
( x > y) 0 else f(x) + sumofnum(x + 1,y,f)}
```

Anonymous Function

```
def sumCubesHOA(a:Int, b: Int) = sum(x=>
x*x*x,1,4)
```

Examples of Higher Oder Functions

Filter

```
(1 to 9).filter(_ % 2 == 0)
```

```
def EvenOdd (x:Int) = { x % 2 == 0}
(1 to 9).filter(EvenOdd)
```

ReduceLeft

```
(1 to 9).reduceLeft(_ * _)
```

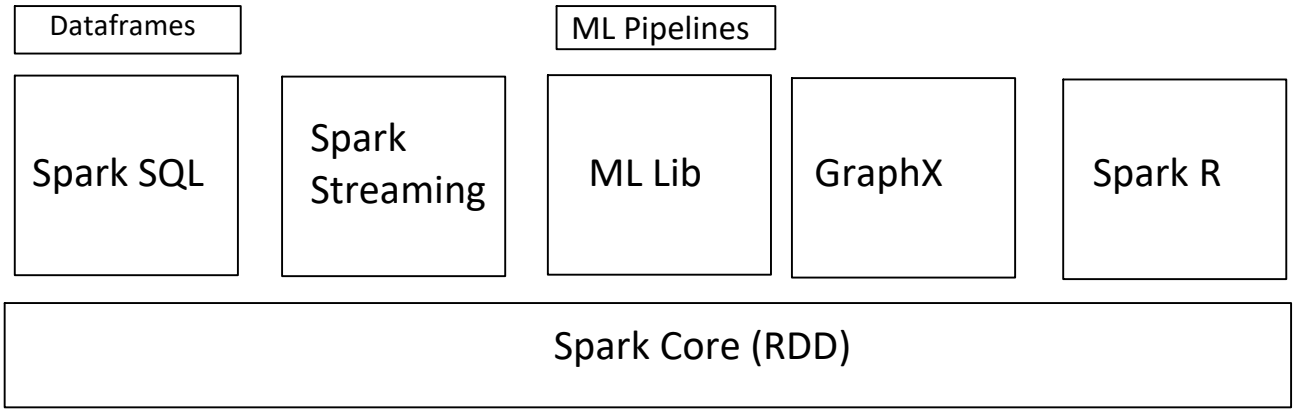
```
def multiply(x :Int, y:Int) = { x * y}

(1 to 9).reduceLeft(multiply)
```

Functions That returns Functions

```
//Higher Order Function That returns Function

  def urlBuilder(ssl: Boolean, domainName:String) :
(String,String) => String = {
    val schema = if (ssl) "https://" else "http://"
    (endpoint:String, query:String) => s"$schema
$domainName/$endpoint?$query" }

  val domainName = "www.example.com"

  val endpoint = "users"
  val query = "id=1"
  def getURL = urlBuilder(ssl=true,domainName)
  val url = getURL(endpoint,query)
  println(url)
```

Spark Components

09 July 2020 01:29



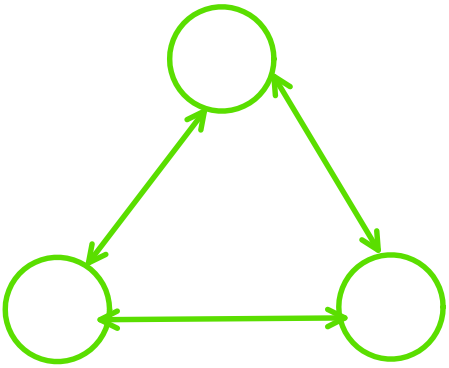
Spark Core - RDD - Using RDDs we can analyse all types of data

Spark SQL - Dataframes - Structured/ Semi Structured Data

Spark Streaming - Dstreams - Analyse Real Time Streaming Data

Spark ML Lib - Pipelines - spark.ml (Dataframes) and spark.mllib(RDD)

Graphx - Graphical Data



Spark R - Leverage R language , can use Spark R

Spark Core - RDD as core abstraction Structuted, SemiStructured
Quasistructured, Unstructured

Spark SQL - Structutred , Semi Structured

Spark Streaming - Real Time Application

MLLib - Machine Learning Problem Statement
Graphx - Analyse Graphical data

Spark R - R analytical Processing

Spark Core - RDD - Resilient Distributed Datasets
Structured, Semi Structured, Quasi, Unstructured

Spark SQL - Replacement for Hive
Structured , Semi Structed

Spark Streaming - Real Time Applications
Dstream
MLLib - spark.ml - Dataframes
Spark.mllib - RDDs

Machine Learning Problem Statements

Graphx = Analyse Graphical Data

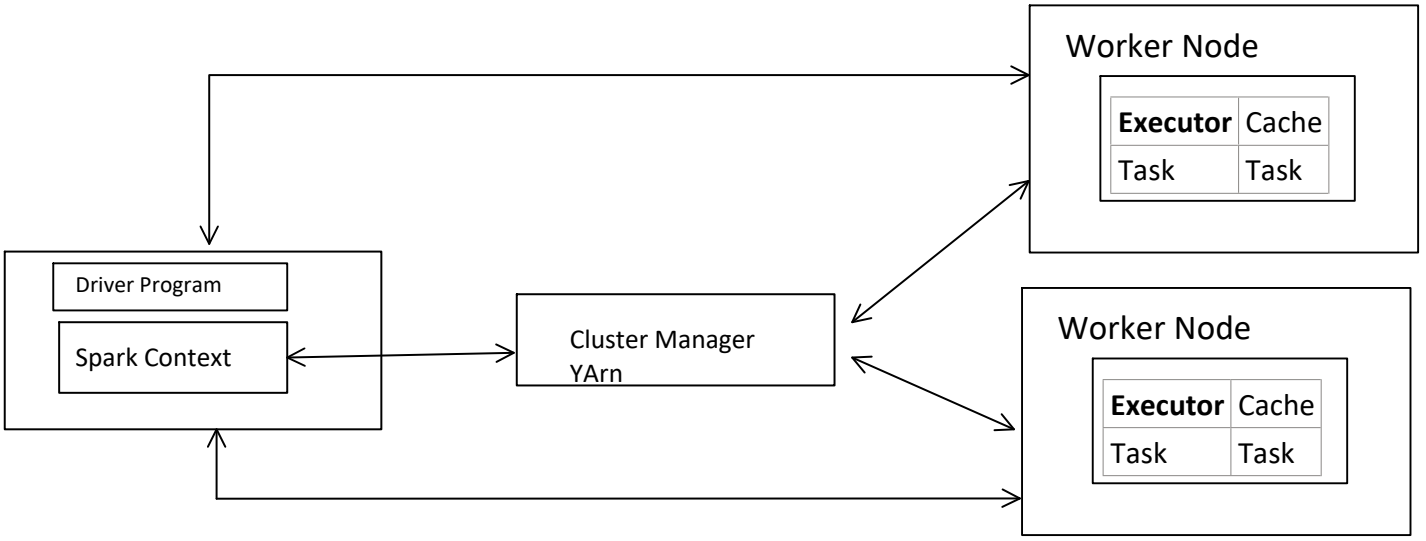
Person - Visit an App - Id1
Website - Id2
In Store - Physical - Id3
Campaign - Id4

Spark R - Leverage the Power of R Language

Go to Slide 22

Spark Architecture

09 July 2020 14:19



I have a locked Door
I want to open it
I need a Key

Key Maker

Spark is Closed Door
Spark Context is the key
Driver Program Creates the Spark Context

Deployment Mode

09 July 2020 23:14

Client Mode - Spark-shell
Cluster Mode -

Go To slide 33