Spark學習前奏 –
Scala基礎教學

Yung-Chuan Lee

前言

- 為降低Spark RDD及MLlib開發學習門檻，本教學整理Spark應用程式開發常用之Scala語法提供學員先備知識
- 本文件為速成捷徑，如想對Scala有較整體的概念，推薦以下教學資源：
  - http://www.codedata.com.tw/java/scala-tutorial-easy-statically-typed-lang/
  - http://openhome.cc/Gossip/Scala/

Scala 語言簡介

- Scala 是 Scalable Language 的意思(可大可小的語言)
- Scala 是 物件導向 和 函數式語言的混合
  - 透過lambda expression讓程式更簡潔

```
Scala: List(1,2,3,4,5).foreach(x=>println("item %d".format(x)))
```

```
Java:
Int[] intArr = new Array[] {1,2,3,4,5};
for (int x: intArr) println(String.format("item %d", x));
```

- scala可叫用Java及.NET龐大的函式庫
- 對平行處理有更佳的支援(函數式語言、actor model、akka)
- Spark 的原生實作語言，支援最完整、文件也最齊全

Scala常用語法整理

- import
  - import org.apache.spark.SparkContext
  - import org.apache.spark.rdd._ (引入rdd底下所有class)
  - import org.apache.spark.mllib.clustering.{ KMeans, KMeansModel } (只引入clustering底下兩個class)
- 變數宣告
  - val int1: Int = 5 (給值後不能再重新給值，否則會error)
  - var int2: Int = 5 (可不斷給新值)
  - val int = 5 (不宣告型別，由編譯器推斷)
- 方法定義(無回傳值)
  - def voidFunc(param1: Type, param2: Type2) = { … }

```
def setLogger = {
    Logger.getLogger("com").setLevel(Level.OFF)
    Logger.getLogger("io").setLevel(Level.OFF)
  }
```

Scala常用語法整理

- 方法定義(單一回傳值)
  - def rtnFunc1(param1: Type, param2: Type2): Type3 = {
    val v1:Type3 = …
    v1 //最後一行為回傳值，型態要和宣告相符
    }
- 方法定義(多回傳值)
  - def rtnFunc2(param1: Type, param2: Type2): (Type3, Type4) = {
    val v1: Type3 = …
    val v2: Type4= …
    (v1, v2)
    //最後一行為回傳值，個數及型態要和宣告相符
    }

```
def getMinMax(intArr: Array[Int]):(Int,Int) = {
    val min=intArr.min
    val max=intArr.max
    (min, max)
  }
```

Scala常用語法整理

- 取得方法執行結果
  - val res = rtnFunc1(param1, param2) (針對單一回傳值方法，res可不宣告回傳型別)
  - val (res1, res2) = rtnFunc2(param1, param2) (針對多回傳值方法，res1,res2可不宣告回傳型別)
  - val (_, res2) = rtnFunc2(param1, param2) (針對多回傳值方法可用底線代表不使用之回傳值)

```
val (min,max)=getMinMax(intArr)
val (_, max)=getMinMax(intArr)
```

- For Loop使用
  - for (i <- collection) { … }
- For Loop使用 (使用yield關鍵字產生集合物件)
  - val rtnArr = for (i <- collection) yield { … }

```
val intArr = Array(1,2,3,4,5,6,7,8,9)
val multiArr=
    for (i <- intArr; j <- intArr)
    yield  { i*j }
//multiArr為長度81的陣列，內容為99乘法表
```

Scala常用語法整理

```
val intArr = Array(1,2,3,4,5,7,8,9)
val res=getMinMax(intArr) //res=(1,9)=>tuple
val min=res._1 //取得res第一個元素
val max=res._2 //取得res第二個元素
```

- Tuple的使用
  - Tuple 是一種固定長度，但元素可以是不同型別的容器。
  - 以val v=(v1,v2,v3...)方式宣告，透過v._1, v._2, v._3…取值
  - 常搭配lambda語法一同使用

```
val intArr = Array((1,2,3),(4,5,6),(7,8,9)) //intArr為Tuple陣列
val intArr2=intArr.map(x=> (x._1 * x._2 * x._3))
//intArr2: Array[Int] = Array(6, 120, 504)
val intArr3=intArr.filter(x=> (x._1 + x._2 > x._3))
//intArr3: Array[(Int, Int, Int)] = Array((4,5,6), (7,8,9))
```

  - 在lambda中常見以底線(_)代表值的方式以簡化寫法

```
val intArr = Array((1,2,3),(4,5,6),(7,8,9)) //intArr為Tuple陣列
def getThird(x:(Int,Int,Int)): Int = { (x._3) }
val intArr2=intArr.map(getThird(_))
val intArr2=intArr.map(x=>getThird(x)) //與上一行相同的對應寫法
//intArr2: Array[Int] = Array(3, 6, 9)
```

Scala常用語法整理

- Class
  - Scala的Class用法和JAVA的Class大致相同，不過寫法可更簡潔
    - 成員變數或方法若沒明確指定飾字（如 private / protected）則預設為public
    - 成員變數可在Class的定義裡直接定義

```
Scala:
class Person(userID: Int, name: String) //定義兩個public的成員變數

class Person(val userID: Int, var name: String)
//定義兩個public的成員變數，其中userID只能給值一次
val person = new Person(102, "John Smith”)//設定成員變數值
person.userID //回傳102
```

```
第一個Person class定義的Java寫法:
public Class Person {
    private final int userID;
    private final String name;
    public Person(int userID, String name) {
        this.userID = userID;
        this.name = name;
    }}
```

Scala常用語法整理

- Object
  - Scala中沒有 static 變數與函式的概念，所有成員及方法均依存於 instance
  - Scala中可透過Object來實作static的用法
    - Scala的Object其實是singleton的class instance
- Scala Object vs Class
  - object用於定義utility或讓Spark執行之Driver Program
  - class用於定義裝載資料之Entity

```
Scala Object:
object Utility {
  def isNumeric(input: String): Boolean = input.trim()
     .matches(s"""[+-]?((\\d+(e\\d+)?[lL]?)|(((\\d+(\\.\\d*)?)|(\\.\\d+))(e\\d+)?[fF]?))""")

  def toDouble(input: String): Double = {
    val rtn = if (input.isEmpty() || !isNumeric(input)) Double.NaN else input.toDouble
    rtn
  }}

val d = Utility.toDouble("20") //可直接使用成員方法，不必先new
```

scala中常用的集合操作

- 宣告集合：
  - val intArr = Array(1,2,3,4,5,7,8,9)
- 集合串接
  - val intArrExtra = intArr ++ Array(0,11,12)
- map: 由現有集合內容建立新的集合
- filter: 取出現有集合中符合特定條件的資料，建立子集合
- join: 整合兩個Map中相同Key的所有資料，建立新的Map
- sortBy、reverse: 依據集合內容進行排序
- take(N): 取出集合中前N筆資料建立新的集合

```
val intArr = Array(1,2,3,4,5,7,8,9)
val intArr2=intArr.map(x => x * 2)
//intArr2: Array[Int] = Array(2, 4, 6, 8, 10, 12, 14, 16, 18)
val intArr3=intArr.filter(x => x > 5)
//intArr3: Array[Int] = Array(6, 7, 8, 9)
val intArr4=intArr.reverse
//intArr4: Array[Int] = Array(9, 8, 7, 6, 5, 4, 3, 2, 1)
```

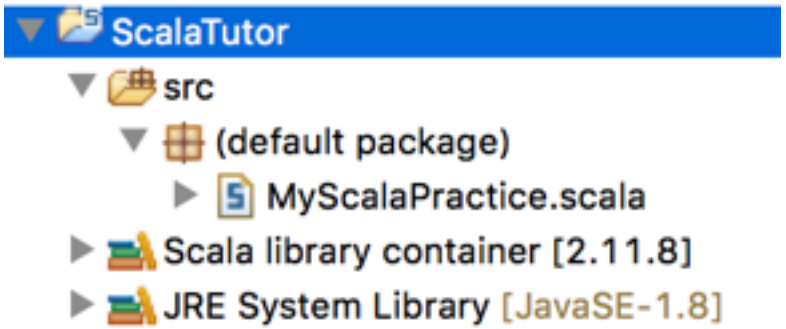
scala中常用的集合操作

- sum: 取得數字集合的總合
  - val sum = Array(1,2,3,4,5,7,8,9).sum
- max: 取得數字集合中的最大值
  - val max = Array(1,2,3,4,5,7,8,9).max
- min: 取得數字集合中的最小值
  - val max = Array(1,2,3,4,5,7,8,9).min
- distinct: 取得集合中，不重覆的元素值

```
val intArr = Array(1,2,3,4,5,7,8,9)
val sum = intArr.sum
//sum = 45
val max = intArr.max
//max = 9
val min = intArr.min
//min = 1
val disc = Array(1,1,1,2,2,2,3,3).distinct
//disc = Array(1,2,3)
```

Scala開發實戰演練

- 請在Scala IDE中建立Scala Project
  - Project名稱：ScalaTutor
  - 在專案中建立一Scala Object(名稱：MyScalaPractice)
- Scala IDE的安裝及設定請參考https://github.com/yclee0418/sparkTeach/blob/master/installation/Spark201-Scala-IDE-installation.pdf (p8~13)



Scala開發實戰演練

- 在MyScalaPractice實作以下邏輯：
  - [Practice1]以case class語法定義一個BikeData Class記錄每小時腳踏車租借量，包含以下成員變數：
    - id(序號):String
    - date(日期):String
    - hour(小時):Int
    - cnt(租借量): Int

```
參考語法：
case  class BikeData(id:???, date:???, hour:???, cnt: ???)
//請自行填入成員變數的型態
```

Scala開發實戰演練

- 在MyScalaPractice實作以下邏輯:
  - 撰寫main方法(參考本文件p17)，作為程式執行入口；並在main方法中實作以下邏輯：
    - [Practice2]建立一個BikeData Array，並加入以下5個BikeData
      - id="1", date="20110101", hour=7, cnt=300
      - id="2", date="20110101", hour=8, cnt=420
      - id="3", date="20110102", hour=6, cnt=340
      - id="4", date="20110102", hour=8, cnt=240
      - id="5", date="20110103", hour=7, cnt=590

參考語法：

```
var bikeArray: Array[BikeData] = Array()
bikeArray ++= Array(BikeData("1", "20110101", 7, 300))
//以下請練習撰寫加入id="2"~"5"的BikeData的邏輯
```

Scala開發實戰演練

- 接續上頁，在main方法中實作以下邏輯：
  - [Practice3]找出租借量(cnt)大於350者，印出bikeData內容
    - 提示：使用filter操作

```
參考語法：
bikeArray.filter { ??? }.foreach(println)
//請填入filter中的篩選語法
```

  - [Practice4]計算bikeArray中cnt的總合
    - 提示：使用map功能將cnt值取成Array[Int]後，呼叫Array[Int]的sum方法取得

```
參考語法：
val sum = bikeArray.map { ??? }.sum
//請填入map中取得cnt變數的語法
```

  - [Practice5]取得bikeArray中cnt的最大及最小值
    - 提示：使用map功能將cnt值取成Array[Int]後，分別呼叫Array[Int]的min及max方法取得

Scala開發實戰演練

- 接續上頁，在main方法中實作以下邏輯：
  - [Practice6]以date欄位為key，grouping計算每日(date)的cnt總合，並以(date, sumOfCnt)格式將結果輸出在console
    - 提示1：以date欄位取distinct，用for迴圈＋filter功能取得每日的bikeData，再用practice4的方法計算每日的sum
    - 提示2：以yield關鍵字，讓for迴圈結果組成Array[(date, sumOfCnt)]

```
參考語法：
val dateDisc = bikeArray.map{ ??? }.distinct
val dateSum =
    for (date <- dateDisc) ??? {
        val sumOfCnt = bikeArray.filter { ??? }.map { ??? }.sum
        (date, sumOfCnt)
      }
dateSum.foreach(println)
//請填入???中的實作
```

Scala開發實戰演練(MyScalaPractice 架構參考)

```
object MyScalaPractice {
  //practice1: 定義case class
  case  class BikeData(id:???, date:???, hour:???, cnt: ???)
  def main(args: Array[String]): Unit = {
    //practice2: 建立BikeData Array，並加入5個element
    var bikeArray: Array[BikeData] = Array()
    bikeArray ++= Array(BikeData("1", "20110101", 7, 300))
    //以下請練習撰寫加入id="2"~"5"的BikeData的邏輯
    bikeArray ++= ……

    //practice3: 找出租借量(cnt)大於350者，印出bikeData內容(限用filter功能)
    println("=====找出租借量(cnt)大於350者=====")
    bikeArray.filter { ??? }.foreach(println)

    //practice4: 計算bikeArray中cnt的總合(提示：使用map功能將cnt值取成Array[Int]後，呼叫Array[Int]的sum方法取得)
    val sum = bikeArray.map { ??? }.sum

    //practice5: 取得bikeArray中cnt的最大及最小值(提示：使用map功能將cnt值取成Array[Int]後，分別呼叫Array[Int]的min及max方法取得)
    val max = ……
    val min = ……
    println("=====cnt的總合：%d, 最大值：%d, 最小值：%d=====".format(sum, max, min))
    //practice6: 以date欄位為key，grouping計算每日(date)的cnt總合，以(date, sumOfCnt)格式列印在console
    //提示：以date欄位取distinct，用for迴圈+filter功能取得每日的bikeData，再用practice4的方法計算每日的sum
    //以yield關鍵字，讓for迴圈結果組成Array[(date, sumOfCnt)]
    val dateDisc = bikeArray.map{ ??? }.distinct
    val dateSum =
      for (date <- dateDisc) ??? {
        val sumOfCnt = bikeArray.filter { ??? }.map { ??? }.sum
        (date, sumOfCnt)
      }
    println("=====以date欄位為key，grouping計算每日(date)的cnt總合=====")
    dateSum.foreach(println)
  }
}
```

Scala開發實戰演練(執行結果輸出參考)

```
=====找出租借量(cnt)大於350者=====
BikeData(2,20110101,8,420)
BikeData(5,20110103,7,590)

=====cnt的總合：1890, 最大值：590, 最小值：240=====

=====以date欄位為key，grouping計算每日(date)的cnt總合=====
(20110101,720)
(20110102,580)
(20110103,590)
```

Scala開發實戰演練(自我挑戰)

- 最後的小驗收，在main方法中實作以下邏輯：
  - [Practice7]以date欄位為key，grouping計算每日(date)的cnt最大值及最小值，並以(date, maxOfCnt, minOfCnt)格式將結果輸出在console
    - 提示1：以date欄位取distinct，用for迴圈＋filter功能取得每日的bikeData，再用practice5的方法取得每日的sum
    - 提示2：以yield關鍵字，讓for迴圈結果組成Array[(date, maxOfCnt, minOfCnt)]