

빅데이터 플랫폼 머신러닝 개발을 위한

# Spark DataFrame APIs & SQL

작성자 : 김진성

# 1. Spark DataFrame/DataSet

---

1. SparkSQL은 구조화된 데이터가 필요
2. SparkML에서도 구조화된 데이터를 다루기 위해서 DataFrame 사용
3. JSON 파일 -> DataFrame 생성
4. RDD -> Dataset, DataFrame 변환 가능
5. DataSet은 여러 개의 row를 갖는 자료구조
  - ✓ DataFrame을 다시 표현한 것(Dataset to represent a Dataframe)
  - ✓ DataSet[Row] = DataFrame
    - ex) val DS = DF.as[(String)]

[Dataset\(Dataframe\) Document](#)

# SparkSession

---

- SparkSession
  - DataFrame/Dataset 생성을 위해서 필요
  - 트랜스포메이션 : DataFrame이나 Dataset 생성
  - 액션 : 연산을 시작하거나 사용 언어에 맞는 데이터 타입으로 변환
  - Spark 2.0 version 필요
  - pom.xml 추가/수정 내용 참고
  - text file은 한 줄을 1 row 처리

`SparkContext(RDD)_vs SparkSession(DataFrame)`

## 2. Spark SQL

---

1. RDD는 데이터에 대한 메타데이터(스키마) 표현 불가능
  - ✓ Spark SQL은 RDD 단점을 보완하기 위한 Spark 모듈
2. SQL을 사용하는 이유
  - ✓ 사용자에게 익숙한 SQL 문법으로 데이터 분석
  - ✓ 맵리듀스보다 빠르고 편리함
3. Spark 2.0 : Spark SQL을 위해서 업데이트(SQL:2003 표준 지원)
  - ✓ 더 많은 쿼리와 파일 포맷 지원 강화(HiveQL 대체)
4. Low-level RDD -> DataSet, DataFrame API로 전환
  - ✓ 머신러닝을 위해서 정형화된 데이터 셋(DataSet, DataFrame) 이용

[SparkSQL Programming Guide](#)

# Spark SQL 사용법

---

- Spark SQL 사용법

1. DataFrame API 이용 : 동적, 가독성 높음

형식) `DF.func().show()`

2. Spark SQL 이용 : SQL문과 유사함, 처리 빠름

형식) `DF.createOrReplaceTempView("table")`

`spark.sql("select * from table").show()`

# 암묵적 변환 방식

- Sacal -> Spark DataFrame 변환 방식

**import spark.implicits.\_**

1) Scala 컬렉션 -> DataFrame 변환

```
val data = List(v1, v2, v3).toDF("eno", "pay", "bonus", "dno")  
Data_df.show()
```

2) case class -> DataFrame 변환

```
val row1 = Dataset("HONG", 35, "회사원", "Male")  
val row2 = Dataset("YOO", 25, "회사원", "Female")  
val row3 = Dataset("LEE", 45, "공무원", "Male")
```

// scala -> DF

```
import spark.implicits._  
val data = List(row1, row2, row3).toDF()  
dataf.show()
```

# Column 사용을 위한 표준내장함수

---

```
import org.apache.spark.sql.functions._
```

형식) `df.select(column` 사용을 위한 표준내장함수)

1. Aggregate functions : max, min, sum, mean, stddev, stddev
2. **Collection functions** : explode, flatten, map\_keys, map\_values
3. **String functions** : split, upper
4. **UDF functions** : udf

참고 사이트

<https://jaceklaskowski.gitbooks.io/mastering-spark-sql/spark-sql-functions.html>

# RDD vs Spark SQL

---

## 1. RDD

- ✓ SparkContext 객체
- ✓ 데이터처리에 필요한 함수를 이용하여 Scala 코드로 작성한 후 이를 RDD의 map, flatMap 등의 메서드로 처리
- ✓ 액션 연산과 트랜스포메이션 연산 제공

## 2. SparkSQL

- ✓ SparkSession 객체
- ✓ DataSet 또는 DataFrame에서 제공하는 Row, Column, 메서드 등을 이용하여 데이터 처리
- ✓ 액션 연산과 트랜스포메이션 연산 제공



## RDD example

```
val sc=new SparkContext(conf) // 컨텍스트 객체
filePath = "src/main/resources"
// RDD 생성 : local file read
val rdd = sc.textFile("src/main/input.txt")
// 단어 수 카운트 : RDD map, flatMap 메서드
rdd.flatMap {
    line.split(" ")
}
.map { word =>
    (word, 1)
}
.reduceByKey(_ + _)
.saveAsTextFile(filePath+"/output") // 외부 저장소 file save
```

## Spark SQL example

```
val spark = SparkSession // 세션 객체
filePath = "src/main/resources"
// DataFrame 생성 : local file read
val df = spark.read.text("src/main/input.txt")
// DataFrame 제공 -> Column, 메서드 이용
val df = df.select(explode(split(col("value"), " ")).as("word"))
// 단어 수 카운트 : DataFrame 제공 메서드 이용
val result = df.groupBy("word").count()

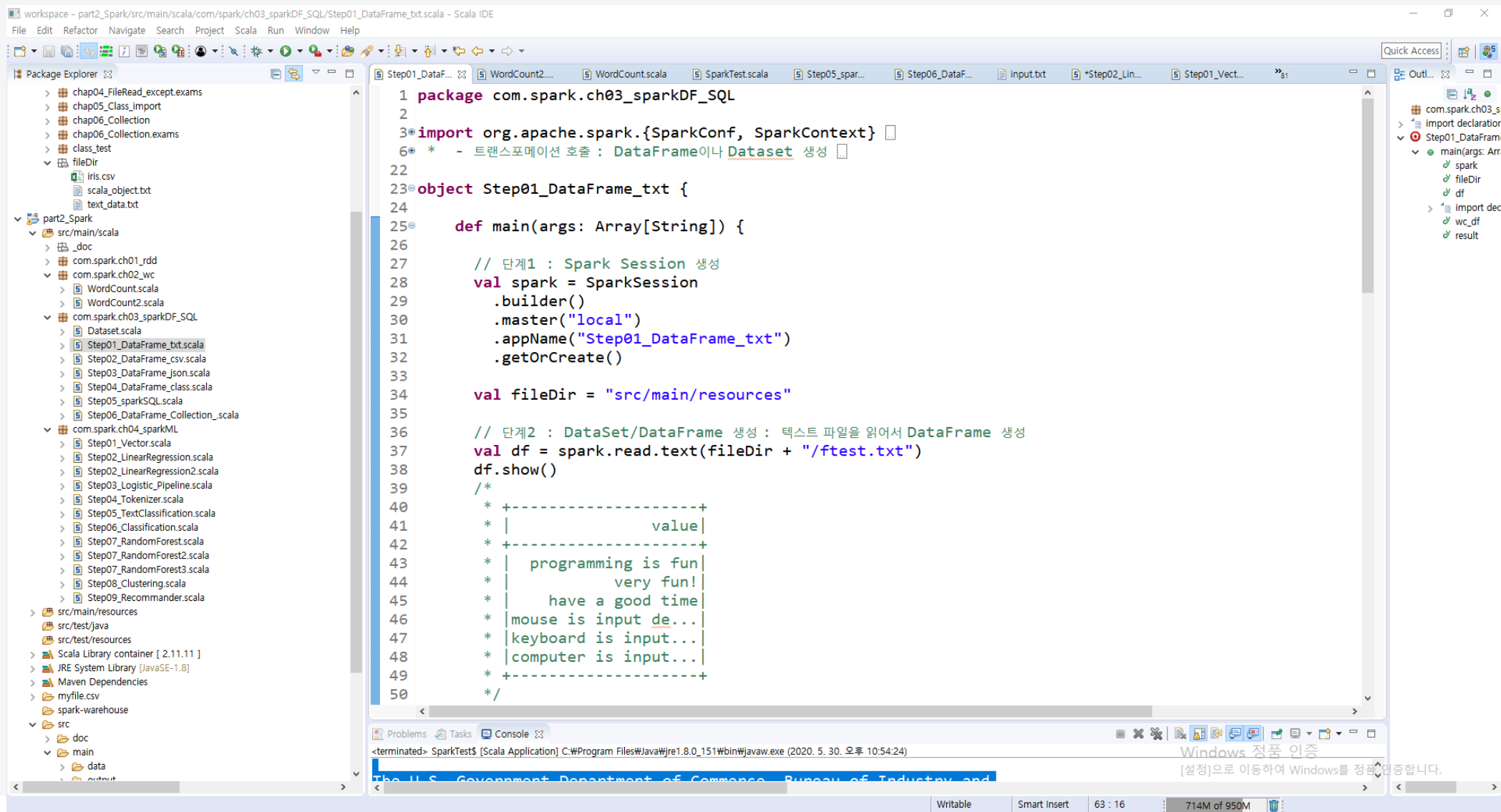
// 외부 저장 file save
result.write.format("csv").option("header", "true")
.save(filePath + "/output")
```

# Spark SQL 작성 단계

---

1. SparkSession 생성 : builder() 메서드 이용
2. DataSet/DataFrame 생성
3. DataSet/DataFrame 이용 데이터 처리(SparkSQL)
4. 처리 결과 외부 저장소(File or HDFS) 저장
5. SparkSession 종료

# Spark DataFrame & SQL example



The screenshot shows an IDE window titled "workspace - part2\_Spark/src/main/scala/com/spark/ch03\_sparkDF\_SQL/Step01\_DataFrame\_txt.scala - Scala IDE". The main editor displays the following Scala code:

```
1 package com.spark.ch03_sparkDF_SQL
2
3 import org.apache.spark.{SparkConf, SparkContext}
4
5 * - 트랜스포메이션 호출 : DataFrame이나 Dataset 생성
6
22
23 object Step01_DataFrame_txt {
24
25   def main(args: Array[String]) {
26
27     // 단계1 : Spark Session 생성
28     val spark = SparkSession
29       .builder()
30       .master("local")
31       .appName("Step01_DataFrame_txt")
32       .getOrCreate()
33
34     val fileDir = "src/main/resources"
35
36     // 단계2 : DataSet/DataFrame 생성 : 텍스트 파일을 읽어서 DataFrame 생성
37     val df = spark.read.text(fileDir + "/ftest.txt")
38     df.show()
39     /*
40     * +-----+
41     * |               value|
42     * +-----+
43     * | programming is fun|
44     * |           very fun!|
45     * |   have a good time|
46     * |mouse is input de...|
47     * |keyboard is input...|
48     * |computer is input...|
49     * +-----+
50     */
51   }
52 }
```

The left sidebar shows the Package Explorer with the following structure:

- workspace
- part2\_Spark
- src/main/scala
- com.spark.ch01\_rdd
- com.spark.ch02\_wc
- com.spark.ch03\_sparkDF\_SQL
- com.spark.ch04\_sparkML
- src/main/resources
- src/test/java
- src/test/resources
- Scala Library container [ 2.11.11 ]
- JRE System Library [JavaSE-1.8]
- Maven Dependencies
- myfile.csv
- spark-warehouse
- src
- doc
- main
- data
- output

The right sidebar shows the Outline view with the following structure:

- com.spark.ch03\_s
- import declaration
- Step01\_DataFrame
- main(args: Arr
- spark
- fileDir
- df
- import dec
- wc\_df
- result

The bottom status bar shows the following information:

- Problems
- Tasks
- Console
- terminated: SparkTest\$ [Scala Application] C:\Program Files\Java\jre1.8.0\_151\bin\javaw.exe (2020. 5. 30. 오후 10:54:24)
- Windows 정품 인증
- [설정]으로 이동하여 Windows를 정품 인증합니다.
- 63 : 16
- 714M of 950M