

Scala for Apache Spark

Markus Dale, medale@asymmetrik.com

Jan 2019

- Slides: <https://github.com/medale/scala-spark/blob/master/presentation/ScalaSpark.pdf>
- Scala Spark Code Examples: <https://github.com/medale/scala-spark>

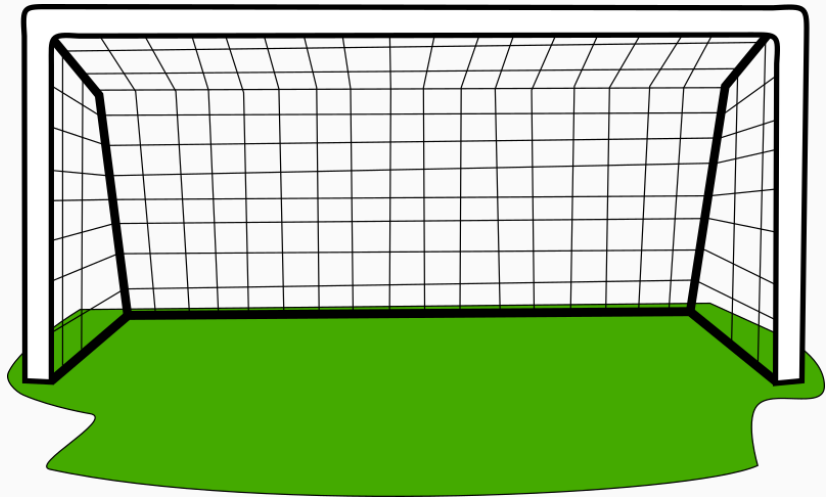


Figure 1: Intro to Scala for Spark

Why Scala for Spark?

- full interoperability with Java
 - strong type system
 - elegant multi-paradigm (functional & OO)
 - less boilerplate/less code
- JVM

```
package com.uebercomputing.scalaspark.common;

public class JavaMain {
    private int answer = 0;
    public JavaMain(int answer) {
        this.answer = answer;
    }
    public int getAnswer() {
        return answer;
    }
    public static void main(String[] args) {
        System.out.println("Starting a Java program...");
        JavaMain jaMain = new JavaMain(42);
        int answer = jaMain.getAnswer();
        System.out.println("The answer was " + answer);
    }
}
```

Scala Main One

```
package com.uebercomputing.scalaspark.common
```

```
class ScalaMainOne(val answer: Int)
```

```
object ScalaMainOne {
```

```
  def main(args: Array[String]): Unit = {  
    println("Starting a Scala program...")
```

```
    val scMain = new ScalaMainOne(42)
```

```
    println(scMain)
```

```
    val answer = scMain.answer
```

```
    println(s"The answer was ${answer}")
```

```
  }
```

```
}
```

Starting a Scala program...

com.uebercomputing.scalaspark.common.ScalaMainOne@256216b3

The answer was 42

Scala Main Two - case class

```
package com.uebercomputing.scalaspark.common
```

```
case class ScalaMainTwo(answer: Int)
```

```
object ScalaMainTwo {
```

```
  def main(args: Array[String]): Unit = {  
    println("Starting a Scala program...")  
    //ScalaMainTwo.apply(42)  
    val scMain = ScalaMainTwo(42)  
    println(scMain)  
    val answer = scMain.answer  
    println(s"The answer was ${answer}")  
  }
```

```
}
```


Starting a Scala program...

ScalaMainTwo(42)

The answer was 42

Scala Main Two - javap ScalaMainTwo.class

```
public class ScalaMainTwo implements Product,Serializable
    public static Option<Object> unapply(ScalaMainTwo);
    public static ScalaMainTwo apply(int);
...
    public ScalaMainTwo copy(int);
...
    public int productArity();
    public Object productElement(int);
    public Iterator<Object> productIterator();
...
    public int hashCode();
    public String toString();
    public boolean equals(Object);
...
```

```
import org.apache.spark.sql.SparkSession

object HelloSparkWorld {
  ...
  def main(args: Array[String]): Unit = {

    val spark = SparkSession.builder.
      appName("HelloSparkWorld").
      master("local[2]").
      getOrCreate()

    process(spark)

    spark.close()
  }
}
```

SparkSession Scala API

sparksession

#ABCDEFGHIJKLMNOPQRSTUVWXYZ – deprecated

display packages only

org.apache.spark.sql

hide focus

SparkSession

SparkSessionExtensions

org.apache.spark.sql

SparkSession

SparkSessionExtensions

org.apache.spark.sql

SparkSession

class **SparkSession** extends Serializable with Closeable with [Logging](#)

The entry point to programming Spark with the Dataset and DataFrame API.
In environments that this has been created upfront (e.g. REPL, notebooks), use the builder to get an existing session:

```
SparkSession.builder().getOrCreate()
```


The builder can also be used to create a new session:

```
SparkSession.builder
  .master("local")
  .appName("Word Count")
  .config("spark.some.config.option", "some-value")
  .getOrCreate()
```

Self Type: [SparkSession](#)
Annotations: [@Stable\(\)](#)
Source: [SparkSession.scala](#)

► Linear Supertypes

Q

Ordering: [Alphabetic](#) [By Inheritance](#)

Inherited: [SparkSession](#) [Logging](#) [Closeable](#) [AutoCloseable](#) [Serializable](#) [Serializable](#) [AnyRef](#) [Any](#)

Hide All [Show All](#)

Visibility: [Public](#) [All](#)

Value Members

► def [baseRelationToDataFrame](#)(baseRelation: [BaseRelation](#)): [DataFrame](#)
Convert a BaseRelation created for external data sources into a DataFrame.

► lazy val [catalog](#): [Catalog](#)
Interface through which the user may create, drop, alter or query underlying databases, tables,

► def [close](#)(): Unit
Synonym for stop().

► lazy val [conf](#): [RuntimeConfig](#)
Runtime configuration interface for Spark.

► def [createDataFrame](#)(data: List[Row], baseClass: Class[Row], DataFormat: DataFormat): DataFrame