Mohamed Camara

Follow
16 Followers
About

Handling Exceptions In Apache Spark

Mohamed Camara Jun 5, 2020 · 2 min read

Sometimes when running a program you may not necessarily know what errors could occur. In such a situation, you may find yourself wanting to catch all possible exceptions. Your end goal may be to save these error messages to a log file for debugging and to send out email notifications. We will see one way how this could possibly be implemented using Spark.

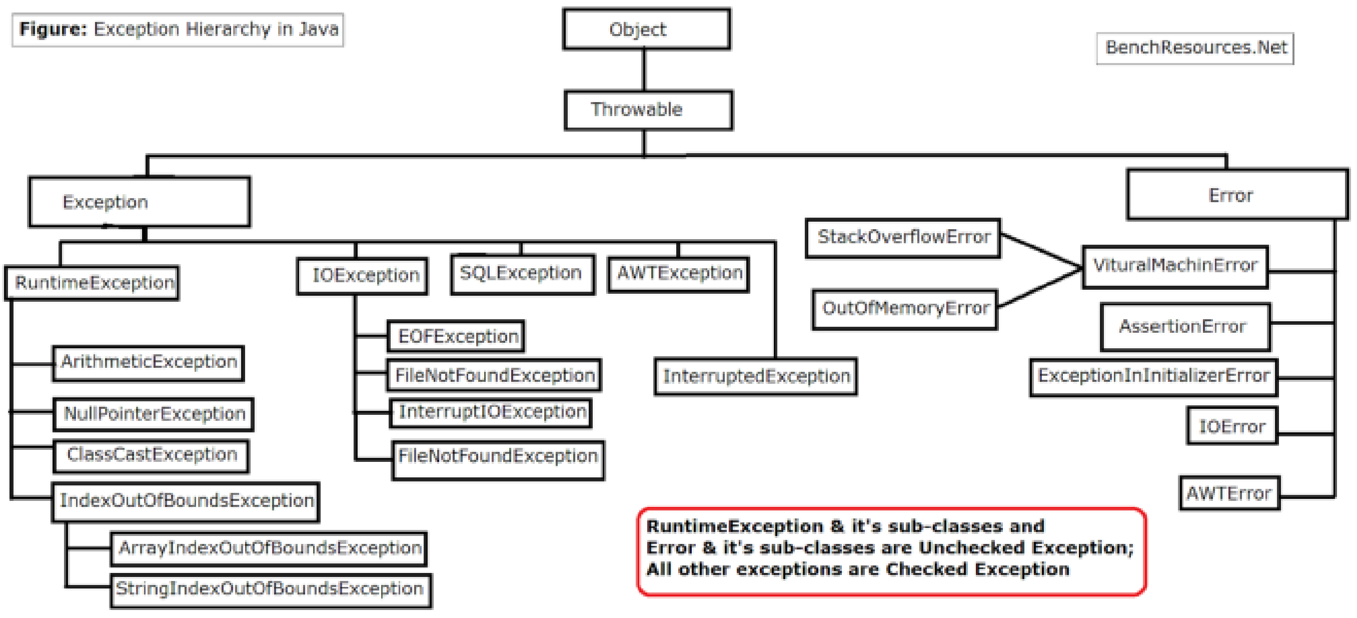
Source:
https://datafloq.com/read/understand-the-fundamentals-of-delta-lake-concept/7610

Scala offers different classes for functional error handling. These classes include but are not limited to *Try/Success/Failure, Option/Some/None, Either/Left/Right*. Depending on

For example, instances of Option result in an instance of either *scala.Some* or *None* and can be used when dealing with the potential of null values or non-existence of values. In other words, a possible scenario would be that with Option[A], some value A is returned, Some[A], or None meaning no value at all. *scala.Option* eliminates the need to check whether a value exists and examples of useful methods for this class would be contains, map or flatmap methods.

Instances of Try, on the other hand, result either in scala.util.Success or scala.util.Failure and could be used in scenarios where the outcome is either an exception or a zero exit status.

We will be using the {Try,Success,Failure} trio for our exception handling.



Source: https://datafloq.com/read/understand-the-fundamentals-of-delta-lake-concept/7610

Only non-fatal exceptions are caught with this combinator. Example of error messages that are not matched are VirtualMachineError (for example, OutOfMemoryError and StackOverflowError, subclasses of VirtualMachineError), ThreadDeath, LinkageError, InterruptedException, ControlThrowable. The Throwable type in Scala is java.lang.Throwable.

```scala
import java.io.{File, PrintWriter, StringWriter}
import scala.util.{Try, Success, Failure}
//import scala.util.control.NonFatal
import java.util.Calendar
import java.text.SimpleDateFormat
import org.apache.spark.sql.DataFrame

//MAIN

     def getFeed (datarecord: String, date_accnt_opened: String): Try[DataFrame] = {

          val sqlContext = new org.apache.spark.sql.hive.HiveContext(sc)
          val erroutput = new StringWriter
          val todays_trans_dt = new SimpleDateFormat("yyyyMMdd").format(Calendar.getInstance.get
          val datarecord = ""
          val date_accnt_opened = ""


          try {
          val testdf = sqlContext.sql(s"""Select * from $datarecord where date_accnt_opened = '$
          testdf.show(3)

          Success(testdf)
         } catch {

           case d: Throwable => d.printStackTrace(new PrintWriter(erroutput))
           new PrintWriter(s"/root/spark_jobs/logs/${datarecord}_$todays_trans_dt.log") //Saves
            {
              write(erroutput.toString);

              close
            }
            //Calls the send email method by passing datarecord as parameter
           sendEmail(datarecord)

            Failure(d)
         }

      }
```

spark_exception_handling_example.scala hosted with ❤ by GitHub view raw harmless Throwables.

Scala Standard Library 2.12.3 - scala.util.Try

Scala Standard Library 2.12.3 - scala.util.Try

Scala Standard Library 2.12.3 - scala.util.Trywww.scala-lang.org

https://docs.scala-lang.org/overviews/scala-book/functional-error-handling.html