

# Java-Success.com

Prepare to fast-track, choose & go places with 800+ Java & Big Data Q&As with lots of code & diagrams.

[Home](#) [Why? ▾](#) [300+ Java FAQs ▾](#) [300+ Big Data FAQs ▾](#) [Courses ▾](#)

[👤 Membership ▾](#) [Your Career ▾](#)

[Home](#) › [bigdata-success.com](#) › [Tutorials - Big Data](#) › [TUT - File Formats](#) › 04: Convert

XML file To an Avro File with Apache Spark – writing & reading

## 04: Convert XML file To an Avro File with Apache Spark – writing & reading

 Posted on [May 19, 2016](#)

This extends [Convert XML file To an Avro File – writing & reading](#).

**Step 1:** The pom.xml file should include the **Apache Spark & Avro** libraries as shown below.

```
1
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
4     <modelVersion>4.0.0</modelVersion>
```

### 300+ Java Interview FAQs

300+ Java FAQs



16+ Java Key Areas Q&As



150+ Java Architect FAQs



80+ Java Code Quality Q&As



150+ Java Coding Q&As



### 300+ Big Data Interview FAQs

300+ Big Data FAQs



Tutorials - Big Data



TUT -  Starting Big Data

TUT - Starting Spark & Scala

```

6      <groupId>my.app</groupId>
7      <artifactId>datawrangling-java</artifactId>
8      <version>0.0.1-SNAPSHOT</version>
9      <packaging>jar</packaging>
10
11     <name>datawrangling-java</name>
12     <url>http://maven.apache.org</url>
13
14     <properties>
15         <maven.compiler.source>1.8</maven.compiler.source>
16         <maven.compiler.target>1.8</maven.compiler.target>
17         <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
18         <junit.version>4.8.1</junit.version>
19         <avro.version>1.7.7</avro.version>
20         <hadoop.version>2.7.2</hadoop.version>
21         <spark.version>1.3.0</spark.version>
22     </properties>
23
24     <dependencies>
25         <dependency>
26             <groupId>junit</groupId>
27             <artifactId>junit</artifactId>
28             <version>${junit.version}</version>
29             <scope>test</scope>
30         </dependency>
31
32         <!-- AVRO -->
33         <dependency>
34             <groupId>org.apache.avro</groupId>
35             <artifactId>avro</artifactId>
36             <version>${avro.version}</version>
37         </dependency>
38         <dependency>
39             <groupId>org.apache.avro</groupId>
40             <artifactId>avro-mapred</artifactId>
41             <version>${avro.version}</version>
42             <classifier>hadoop2</classifier>
43             <exclusions>
44                 <exclusion>
45                     <groupId>org.apache.avro</groupId>
46                     <artifactId>*</artifactId>
47                 </exclusion>
48             </exclusions>
49         </dependency>
50
51         <!-- Hadoop -->
52         <dependency>
53             <groupId>org.apache.hadoop</groupId>
54             <artifactId>hadoop-hdfs</artifactId>
55             <version>${hadoop.version}</version>
56             <exclusions>
57                 <exclusion>
58                     <groupId>javax.servlet</groupId>
59                     <artifactId>*</artifactId>
60                 </exclusion>

```

TUT - Starting with Python

TUT - Kafka

TUT - Pig

TUT - Apache Storm

TUT - Spark Scala on Zeppelin

TUT - Cloudera

TUT - Cloudera on Docker

TUT - File Formats

TUT - Spark on Docker

TUT - Flume

TUT - Hadoop (HDFS)

TUT - HBase (NoSQL)

TUT - Hive (SQL)

TUT - Hadoop & Spark

TUT - MapReduce

TUT - Spark and Scala

TUT - Spark & Java

TUT - PySpark on Databricks

TUT - Zookeeper

## 800+ Java Interview Q&As

300+ Core Java Q&As



300+ Enterprise Java Q&As



150+ Java Frameworks Q&As



120+ Companion Tech Q&As



Tutorials - Enterprise Java



```

61         </exclusions>
62     </dependency>
63     <dependency>
64         <groupId>org.apache.hadoop</groupId>
65         <artifactId>hadoop-client</artifactId>
66         <version>${hadoop.version}</version>
67         <exclusions>
68             <exclusion>
69                 <groupId>javax.servlet</groupId>
70                 <artifactId>*</artifactId>
71             </exclusion>
72         </exclusions>
73     </dependency>
74
75     <!-- XML -->
76     <dependency>
77         <groupId>org.eclipse.persistence</groupId>
78         <artifactId>org.eclipse.persistence.mo<
79         <version>2.6.2</version>
80     </dependency>
81
82     <!-- Spark -->
83     <dependency>
84         <groupId>org.apache.spark</groupId>
85         <artifactId>spark-core_2.11</artifactId>
86         <version>${spark.version}</version>
87         <exclusions>
88             <exclusion>
89                 <groupId>javax.servlet</groupId>
90                 <artifactId>*</artifactId>
91             </exclusion>
92         </exclusions>
93     </dependency>
94 </dependencies>
95 </project>
96

```

**Step 2:** The **report.xml** file under  
“src/main/resources/data”.

```

1
2 <?xml version="1.0" encoding="UTF-8"?>
3 <transactionReports xmlns="http://mytutorial.com/
4     <transactionReport>
5         <report>
6             <reportNumber>9999</reportNumber>
7             <createdDatetime>2015-06-15T11:29:52+1
8             <processedDatetime>2015-06-15T11:29:52
9             <reportStatusCode>Active</reportStatus
10         </report>
11     </transactionReport>

```

```
12 </transactionReports>
13
```

**Step 3:** The avro schema file “**trans-report.avsc**” under “src/main/resources/schema”.

```
1
2 {"namespace": "mytutorial.com.report",
3  "type": "record",
4  "name": "ReportAvro",
5  "fields": [
6    {"name": "reportNumber", "type": "string"},
7    {"name": "createdDatetime", "type": "string"},
8    {"name": "processedDatetime", "type": "string"},
9    {"name": "reportStatusCode", "type": "string"}
10 ]
11 }
12
```

**Step 4:** The **Report.java** to map XML contents to POJO (Plain Old Java Object).

```
1
2 package com.mytutorial.pojo;
3
4 public class Report {
5
6     private String reportNumber;
7     private String createdDatetime;
8     private String processedDatetime;
9     private String reportStatusCode;
10
11     public Report(String reportNumber, String createdDatetime, String processedDatetime, String reportStatusCode) {
12         this.reportNumber = reportNumber;
13         this.createdDatetime = createdDatetime;
14         this.processedDatetime = processedDatetime;
15         this.reportStatusCode = reportStatusCode;
16     }
17
18     public String getReportNumber() {
19         return reportNumber;
20     }
21
22     public void setReportNumber(String reportNumber) {
23         this.reportNumber = reportNumber;
24     }
25
```

```
26     public String getCreatedDatetime() {
27         return createdDatetime;
28     }
29
30     public void setCreatedDatetime(String createdDatetime) {
31         this.createdDatetime = createdDatetime;
32     }
33
34     public String getProcessedDatetime() {
35         return processedDatetime;
36     }
37
38     public void setProcessedDatetime(String processedDatetime) {
39         this.processedDatetime = processedDatetime;
40     }
41
42     public String getReportStatusCode() {
43         return reportStatusCode;
44     }
45
46     public void setReportStatusCode(String reportStatusCode) {
47         this.reportStatusCode = reportStatusCode;
48     }
49
50     @Override
51     public String toString() {
52         return "Report [reportNumber=" + reportNumber + ", processedDatetime=" + processedDatetime + ", reportStatusCode=" + reportStatusCode + "]";
53     }
54 }
55 }
56 }
```

### Step 5: Finally, the stand-alone

“ConvertXmlToSequenceWithSpark” to convert an XML to POJO, and then to AVRO “GenericRecord”, and then to an AVRO file “data/report.avro”.

```
1
2 package com.mytutorial;
3
4 import java.io.File;
5 import java.io.IOException;
6 import java.io.StringReader;
7 import java.net.URL;
8 import java.util.ArrayList;
9 import java.util.Iterator;
10 import java.util.List;
11
12 import javax.xml.namespace.NamespaceContext;
13 import javax.xml.xpath.XPath;
```

```
14 import javax.xml.xpath.XPathConstants;
15 import javax.xml.xpath.XPathExpressionException;
16 import javax.xml.xpath.XPathFactory;
17
18 import org.apache.avro.Schema;
19 import org.apache.avro.generic.GenericData;
20 import org.apache.avro.generic.GenericRecord;
21 import org.apache.avro.mapred AvroKey;
22 import org.apache.avro.mapreduce AvroJob;
23 import org.apache.avro.mapreduce AvroKeyInputFormat;
24 import org.apache.avro.mapreduce AvroKeyOutputFormat;
25 import org.apache.commons.io.FileUtils;
26 import org.apache.hadoop.conf.Configuration;
27 import org.apache.hadoop.io.NullWritable;
28 import org.apache.hadoop.mapreduce.Job;
29 import org.apache.spark.SparkConf;
30 import org.apache.spark.api.java.JavaPairRDD;
31 import org.apache.spark.api.java.JavaSparkContext;
32 import org.w3c.dom.Node;
33 import org.xml.sax.InputSource;
34
35 import scala.Tuple2;
36 import com.mytutorial.pojo.Report;
37
38 public class ConvertXmlToAvroWithSpark {
39
40     private static final String FILE_IN_PATH = "C:\\Users\\Administrator\\Desktop\\xml\\input.xml";
41     private static final String FILE_OUT_PATH = "C:\\Users\\Administrator\\Desktop\\xml\\output.avro";
42     private static final String AVRO_SCHEMA_FILE = "C:\\Users\\Administrator\\Desktop\\xml\\schema.avsc";
43
44     final static JavaSparkContext sc;
45
46     static {
47         SparkConf conf = new SparkConf()
48             .setAppName("Sequence To Avro")
49             .setMaster("local[1]")
50             .set("spark.executor.memory", "1g")
51             .set("spark.serializer",
52                 "org.apache.spark.serializer.KryoSerializer");
53         sc = new JavaSparkContext(conf);
54     }
55
56     public static void main(String[] args) throws XPathExpressionException {
57         URL resource = ConvertXmlToSequenceWithSpark.class.getResource(FILE_IN_PATH);
58
59         File inputFile = new File(resource.getPath());
60         File outputFile = new File(resource.getPath().replace(FILE_IN_PATH, FILE_OUT_PATH));
61         File avroSchemaFile = new File(resource.getPath().replace(FILE_IN_PATH, AVRO_SCHEMA_FILE));
62
63         Report report = convertXmlToPojo(inputFile);
64
65         write(avroSchemaFile, outputFile, report);
66
67     }
68 }
```

```

69         read(outputFile);
70
71         sc.stop();
72     }
73
74     public static void write(File avroSchemaFile
75         throws IOException {
76         Schema avroSchema = new Schema.Parser().parse(avroSchemaFile);
77         GenericRecord myrecord = new GenericData.Record(avroSchema);
78         myrecord.put("reportNumber", report.getReportNumber());
79         myrecord.put("createdDatetime", report.getCreatedDatetime());
80         myrecord.put("processedDatetime", report.getProcessedDatetime());
81         myrecord.put("reportStatus", report.getReportStatus());
82         myrecord.put("reportStatusCode", report.getReportStatusCode());
83
84         Job job = Job.getInstance();
85         AvroJob.setOutputKeySchema(job, avroSchema);
86
87         ArrayList<Tuple2<AvroKey<GenericRecord>, GenericRecord>>
88         arrayList = new ArrayList<>();
89         arrayList.add(new Tuple2<AvroKey<GenericRecord>, GenericRecord>(
90             myrecord, myrecord));
91
92         JavaPairRDD<AvroKey<GenericRecord>, GenericRecord> rdd =
93             rdd.parallelizePairs(arrayList);
94
95         if (outputFile.exists()) {
96             FileUtils.deleteQuietly(outputFile);
97         }
98
99         rdd.saveAsNewAPIHadoopFile(outputFile.getAbsolutePath(),
100             NullWritable.class, AvroKeyOutputFormat.class,
101             job.getConfiguration());
102     }
103
104     public static void read(File avroFileToRead)
105     {
106         @SuppressWarnings("unchecked")
107         JavaPairRDD<AvroKey<GenericRecord>, GenericRecord> rdd =
108             rdd.newAPIHadoopFile(avroFileToRead.getAbsolutePath(),
109                 AvroKeyInputFormat.class, GenericRecord.class,
110                 NullWritable.class, new Configuration());
111
112         List<AvroKey<GenericRecord>> avroKeys = rdd.keys().collect();
113         AvroKey<GenericRecord> avroKey = avroKeys.get(0);
114
115         System.out.println("Avro Data=" + avroKey.get().toString());
116     }
117
118     // Xpath to read XML & convert it to a pojo
119     private static Report convertXmlToPojo(File xmlFile)
120         throws XPathExpressionException {
121         XPath xPath = XPathFactory.newInstance().newXPath();
122         // namespace
123         NamespaceContext ctx = new NamespaceContext() {

```

```
124         public String getNamespaceURI(String prefix) {
125             return prefix.equals("urn") ? "http://www.w3.org/2001/XMLSchema-instance"
126                 : null;
127         }
128
129         public Iterator<String> getPrefixes(String namespaceURI) {
130             return null;
131         }
132
133         public String getPrefix(String uri) {
134             return null;
135         }
136     };
137
138     XPath.setNamespaceContext(ctx);
139     String str = FileUtils.readFileToString(xmlFile);
140     StringReader sr = new StringReader(str);
141     InputSource source = new InputSource(sr);
142
143     // get the DOM
144     Node root = (Node) XPath.evaluate("/", source);
145
146     // use the DOM
147     String reportNumber = XPath.evaluate("//urn:report/urn:reportNumber", root);
148     String createdDatetime = XPath.evaluate("//urn:report/urn:createdDatetime", root);
149     String processedDatetime = XPath.evaluate("//urn:report/urn:processedDatetime", root);
150     String reportStatusCode = XPath.evaluate("//urn:report/urn:reportStatusCode", root);
151
152     return new Report(reportNumber, createdDatetime, processedDatetime, reportStatusCode);
153 }
154 }
```

## Output

```
1
2 Avro Data={"reportNumber": "9999", "createdDatetime": "2022-04-13T07:33:00Z", "processedDatetime": "2022-04-13T07:33:00Z", "reportStatusCode": "SUCCESS"}
3
```

**Note:** The avro file by the Spark job will be written in

**Folder:** /home/cloudera/projects/sequence-file/target/classes/data/report.avro



## File(s):

1	
2	-rw-r--r-- 1 cloudera cloudera 451 May 18 08:46 pa
3	-rw-r--r-- 1 cloudera cloudera 0 May 18 08:46 _S
4	

◀ 12 Apache Spark getting started interview Q&As

05: 7 Java FP (lambda expressions) real life examples in wrangling normal  
& big data ▶

## Disclaimer

The contents in this Java-Success are copyrighted and from EmpoweringTech pty ltd. The EmpoweringTech pty ltd has the right to correct or enhance the current content without any prior notice. These are general advice only, and one needs to take his/her own circumstances into consideration. The EmpoweringTech pty ltd will not be held liable for any damages caused or alleged to be caused either directly or indirectly by these materials and resources. Any trademarked names or labels used in this blog remain the property of their respective trademark owners. Links to external sites do not imply endorsement of the linked-to sites. [Privacy Policy](#)