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/P
4     <modelVersion>4.0.0</modelVersion>
5
```

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

```
    <groupId>my.app</groupId>
    <artifactId>datawrangling-java</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>datawrangling-java</name>
    <url>http://maven.apache.org</url>

    <properties>
        <maven.compiler.source>1.8</maven.compiler
        <maven.compiler.target>1.8</maven.compiler
        <project.build.sourceEncoding>UTF-8</proje
        <junit.version>4.8.1</junit.version>
        <avro.version>1.7.7</avro.version>
        <hadoop.version>2.7.2</hadoop.version>
        <spark-version>1.3.0</spark-version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>${junit.version}</version>
            <scope>test</scope>
        </dependency>

        <!-- AVRO -->
        <dependency>
            <groupId>org.apache.avro</groupId>
            <artifactId>avro</artifactId>
            <version>${avro.version}</version>
        </dependency>
         <dependency>
            <groupId>org.apache.avro</groupId>
            <artifactId>avro-mapred</artifactId>
            <version>${avro.version}</version>
            <classifier>hadoop2</classifier>
            <exclusions>
                <exclusion>
                    <groupId>org.apache.avro</grou
                    <artifactId>*</artifactId>
                </exclusion>
            </exclusions>
        </dependency>

        <!-- Hadoop -->
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-hdfs</artifactId>
            <version>${hadoop.version}</version>
            <exclusions>
                <exclusion>
                    <groupId>javax.servlet</group
                    <artifactId>*</artifactId>
                </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

```
            </exclusions>
        </dependency>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-client</artifactId>
            <version>${hadoop.version}</version>
            <exclusions>
                <exclusion>
                    <groupId>javax.servlet</groupI
                    <artifactId>*</artifactId>
                </exclusion>
            </exclusions>
        </dependency>

        <!-- XML -->
        <dependency>
            <groupId>org.eclipse.persistence</grou
            <artifactId>org.eclipse.persistence.m
            <version>2.6.2</version>
        </dependency>

        <!-- Spark -->
        <dependency>
            <groupId>org.apache.spark</groupId>
            <artifactId>spark-core_2.11</artifactI
            <version>${spark-version}</version>
            <exclusions>
                <exclusion>
                    <groupId>javax.servlet</groupI
                    <artifactId>*</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
    </dependencies>
</project>

```

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

```

<?xml version="1.0" encoding="UTF-8"?>
<transactionReports xmlns="http://mytutorial.com/r
    <transactionReport>
        <report>
            <reportNumber>9999</reportNumber>
            <createdDatetime>2015-06-15T11:29:52+1
            <processedDatetime>2015-06-15T11:29:52
            <reportStatusCode>Active</reportStatus
        </report>
    </transactionReport>

```

```
</transactionReports>

```

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

```

{"namespace": "mytutorial.com.report",
 "type": "record",
 "name": "ReportAvro",
 "fields": [
     {"name": "reportNumber", "type": "string"},
     {"name": "createdDatetime", "type": "string"
     {"name": "processedDatetime", "type": "string
     {"name": "reportStatusCode", "type": "string
 ]
}

```

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

```

package com.mytutorial.pojo;

public class Report {

    private String reportNumber;
    private String createdDatetime;
    private String processedDatetime;
    private String reportStatusCode;

    public Report(String reportNumber, String cre
        this.reportNumber = reportNumber;
        this.createdDatetime = createdDatetime;
        this.processedDatetime = processedDatetim
        this.reportStatusCode = reportStatusCode;
    }

    public String getReportNumber() {
        return reportNumber;
    }

    public void setReportNumber(String reportNumb
        this.reportNumber = reportNumber;
    }


```

```
26      public String getCreatedDatetime() {
27          return createdDatetime;
28      }
29
30      public void setCreatedDatetime(String createdD
31          this.createdDatetime = createdDatetime;
32      }
33
34      public String getProcessedDatetime() {
35          return processedDatetime;
36      }
37
38      public void setProcessedDatetime(String proces
39          this.processedDatetime = processedDatetime
40      }
41
42      public String getReportStatusCode() {
43          return reportStatusCode;
44      }
45
46      public void setReportStatusCode(String reportS
47          this.reportStatusCode = reportStatusCode;
48      }
49
50      @Override
51      public String toString() {
52          return "Report [reportNumber=" + reportNum
53                  + processedDatetime + ", reportSta
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.AvroKeyInputFor
24 import org.apache.avro.mapreduce.AvroKeyOutputFo
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.JavaSparkContex
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 = "
41     private static final String FILE_OUT_PATH =
42     private static final String AVRO_SCHEMA_FILE
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", "1
51                 .set("spark.serializer",
52                         "org.apache.spark.serial
53         sc = new JavaSparkContext(conf);
54     }
55
56     public static void main(String[] args) throw
57             XPathExpressionException {
58         URL resource = ConvertXmlToSequenceWithS
59
60         File inputFile = new File(resource.getPa
61         File outputFile = new File(resource.getP
62         File avroSchemaFile = new File(resource.
63                 + AVRO_SCHEMA_FILE);
64
65         Report report = convertXmlToPojo(inputFi
66
67         write(avroSchemaFile, outputFile, report
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().
77          GenericRecord myrecord = new GenericData
78          myrecord.put("reportNumber", report.getR
79          myrecord.put("createdDatetime", report.g
80          myrecord.put("processedDatetime", report
81                  .toString());
82          myrecord.put("reportStatusCode", report.
83
84          Job job = Job.getInstance();
85          AvroJob.setOutputKeySchema(job, avroSche
86
87          ArrayList<Tuple2<AvroKey<GenericRecord>,
88          arrayList.add(new Tuple2<AvroKey<Generic
89                  myrecord), null));
90
91          JavaPairRDD<AvroKey<GenericRecord>, Void
92                  .parallelizePairs(arrayList);
93
94          if (outputFile.exists()) {
95              FileUtils.deleteQuietly(outputFile);
96          }
97
98          hadoopRDD.saveAsNewAPIHadoopFile(outputF
99                  NullWritable.class, AvroKeyOutpu
100                 job.getConfiguration());
101
102     }
103
104     public static void read(File avroFileToRead)
105
106         @SuppressWarnings("unchecked")
107         JavaPairRDD<AvroKey<GenericRecord>, Null
108                 .newAPIHadoopFile(avroFileToRead
109                         AvroKeyInputFormat.class
110                         NullWritable.class, new
111
112         List<AvroKey<GenericRecord>> avroKeys =
113         AvroKey<GenericRecord> avroKey = avroKey
114
115         System.out.println("Avro Data=" + avroKe
116     }
117
118     // Xpath to read XML & convert it to a pojo
119     private static Report convertXmlToPojo(File
120             XPathExpressionException {
121         XPath xPath = XPathFactory.newInstance()
122         // namespace
123         NamespaceContext ctx = new NamespaceCont
```

```
124             public String getNamespaceURI(String
125                 return prefix.equals("urn") ? "h
126                         : null;
127             }
128
129             public Iterator<String> getPrefixes(S
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(f
140         StringReader sr = new StringReader(str);
141         InputSource source = new InputSource(sr)
142
143         // get the DOM
144         Node root = (Node) xPath.evaluate("/", so
145
146         // use the DOM
147         String reportNumber = xPath.evaluate("//u
148                 root);
149         String createdDatetime = xPath.evaluate(
150                 "//urn:report/urn:createdDatetime
151         String processedDatetime = xPath.evaluate
152                 "//urn:report/urn:processedDatet
153         String reportStatusCode = xPath.evaluate(
154                 "//urn:report/urn:reportStatusCo
155
156         return new Report(reportNumber, createdDa
157                 reportStatusCode);
158     }
159 }
160
```

Output

```
1
2 Avro Data={"reportNumber": "9999", "createdDatetime
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):

```
-rw-r--r-- 1 cloudera cloudera 451 May 18 08:46 pa
-rw-r--r-- 1 cloudera cloudera   0 May 18 08:46 _S
```

‹ 12 Apache Spark getting started interview Q&As

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

Disclaimer