

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](#) > 01a:

Convert XML file To Sequence File – writing & reading – Local File System

01a: Convert XML file To Sequence File – writing & reading – Local File System

 Posted on [May 1, 2016](#)

Sequence files are good for saving **raw data** into HDFS. Sequence files are **compressible** and **splittable**. It is also useful for combining a number of smaller files into a single say 64MB or larger sequence file as HDFS is more suited for larger files. Text files like csv, xml, json, etc can be stored on HDFS in **sequence file format**.

Step 1: Create a Java project using Maven.

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

```

1
2 mvn archetype:generate -DgroupId=com.mytutorial -D
3

```

Step 2: Import it into eclipse as a an “existing maven project”.

Step 3: Add the hadoop dependencies to the **pom.xml** file.

```

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>
5   <groupId>com.mytutorial</groupId>
6   <artifactId>sequence-file</artifactId>
7   <packaging>jar</packaging>
8   <version>1.0-SNAPSHOT</version>
9   <name>sequence-file</name>
10  <url>http://maven.apache.org</url>
11
12  <properties>
13    <maven.compiler.source>1.8</maven.compiler.source>
14    <maven.compiler.target>1.8</maven.compiler.target>
15    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
16    <junit.version>4.8.1</junit.version>
17    <hadoop.version>2.7.2</hadoop.version>
18  </properties>
19
20  <dependencies>
21    <!-- JUnit -->
22    <dependency>
23      <groupId>junit</groupId>
24      <artifactId>junit</artifactId>
25      <version>${junit.version}</version>
26      <scope>test</scope>
27    </dependency>
28
29    <!-- Hadoop -->
30    <dependency>
31      <groupId>org.apache.hadoop</groupId>
32      <artifactId>hadoop-hdfs</artifactId>
33      <version>${hadoop.version}</version>
34      <exclusions>
35        <exclusion>
36          <groupId>javax.servlet</groupId>
37          <artifactId>*</artifactId>
38        </exclusion>
39      </exclusions>

```

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



```
40     </dependency>
41     <dependency>
42         <groupId>org.apache.hadoop</groupId>
43         <artifactId>hadoop-client</artifactId>
44         <version>${hadoop.version}</version>
45         <exclusions>
46             <exclusion>
47                 <groupId>javax.servlet</groupId>
48                 <artifactId>*</artifactId>
49             </exclusion>
50         </exclusions>
51     </dependency>
52
53 </dependencies>
54
55 </project>
56
```

Step 4: Create **src/main/resources** source folder, and under that the “data” sub-folder. Create “**report.xml**”, which needs to be converted to a **sequence file format**.

```
1
2 <?xml version="1.0" encoding="UTF-8"?>
3 <transactionReports xmlns="http://mytutorial.com/transactionReports" >
4     <transactionReport>
5         <report>
6             <reportNumber>9999</reportNumber>
7             <createdDatetime>2015-06-15T11:29:52+05:30</createdDatetime>
8             <processedDatetime>2015-06-15T11:29:52+05:30</processedDatetime>
9             <reportStatusCode>Active</reportStatusCode>
10        </report>
11    </transactionReport>
12 </transactionReports>
13
```

Step 5: Write a stand-alone Java class to convert above report.xml file to report.seq. Write and then read. This example uses the “**LOCAL FILE SYSTEM**”

```
1
2 package com.mytutorial;
3
4 import java.io.File;
5 import java.io.IOException;
```

```
6 import java.net.URL;
7 import java.util.List;
8 import java.util.stream.Collectors;
9
10 import org.apache.commons.io.FileUtils;
11 import org.apache.hadoop.conf.Configuration;
12 import org.apache.hadoop.fs.Path;
13 import org.apache.hadoop.io.BytesWritable;
14 import org.apache.hadoop.io.IOUtils;
15 import org.apache.hadoop.io.IntWritable;
16 import org.apache.hadoop.io.SequenceFile;
17 import org.apache.hadoop.util.ReflectionUtils;
18
19 public class ConvertXmlToSequence {
20     private static final String FILE_IN_PATH = "d
21     private static final String FILE_OUT_PATH = "
22
23     public static void main(String[] args) throws
24         URL resource = ConvertXmlToSequence.class
25
26         Configuration conf = new Configuration();
27
28         File inputFile = new File(resource.getPat
29         Path outputFile = new Path(resource.getPa
30
31         write(conf, inputFile, outputFile); // wr
32         read(conf, outputFile); // read seq file
33     }
34
35     /**
36      * Write a text file to sequence file
37      *
38      * @param conf
39      * @param inputFile
40      * @param outputFile
41      */
42     public static void write(Configuration conf, I
43         SequenceFile.Writer writer = null;
44
45         try {
46             writer = SequenceFile.createWriter(con
47                 SequenceFile.Writer.compressio
48                 SequenceFile.Writer.keyClass(C
49                 SequenceFile.Writer.valueClass
50
51             List<String> lines = FileUtils.readLin
52             String xmlString = lines.stream().map
53             IntWritable key = new IntWritable(1);
54             BytesWritable value = new BytesWritab
55             writer.append(key, value);
56
57         } catch (IOException e) {
58             System.out.println("Error writing: "
59         }
60
```

```
61         finally {
62             IOUtils.closeStream(writer);
63         }
64     }
65
66     /**
67     * Read a sequence file
68     *
69     * @param conf
70     * @param sequenceFileToRead
71     */
72     public static void read(Configuration conf, Path sequenceFileToRead) {
73         SequenceFile.Reader reader = null;
74         try {
75             reader = new SequenceFile.Reader(conf, new IntWritable(), new BytesWritable(), sequenceFileToRead);
76             IntWritable keyRead = (IntWritable) reader.getKey();
77             BytesWritable valueRead = (BytesWritable) reader.getValue();
78             while (reader.next(keyRead, valueRead)) {
79                 System.out.println("key : " + keyRead + " value : " + valueRead);
80                 valueRead = (BytesWritable) reader.getValue();
81             }
82         } catch (IOException e) {
83             System.out.println("Cannot read sequence file");
84         }
85
86         IOUtils.closeStream(reader);
87     }
88 }
89
```

Output:

```
1
2 key : 1 - value : <?xml version="1.0" encoding="UTF-8" standalone="no" ?>
3
```

As you can see, the sequence files are stored as key/value pairs. The **key** being IntWritable "1" and the **value** being the "whole XML".

The **report.seq** file gets written to the project folder's "**target/classes/data**" folder. The "value" will be stored as binary.

Step 6: If you want to open the sequence file in eclipse, you need to install the hex editor plugin from

"http://ehp.sourceforge.net/update".

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ASCII
00000000	12	45	11	05	10	67	72	67	2E	61	70	61	65	6A	65	2E	hadoop._seg.apache.
00000001	68	61	64	67	67	70	2E	69	67	2E	49	6E	74	57	72	69	hadoop.io.InvM1
00000002	74	61	65	65	65	67	72	67	72	74	69	61	69	6A	65	65	table=org.apache
00000003	2E	6A	61	64	67	70	2E	69	67	2E	42	42	79	74	65	73	hadoop.io.Bytes
00000004	57	72	69	67	74	61	62	6C	65	00	00	00	00	00	00	00	Writable.....
00000005	79	6E	67	72	67	68	69	69	69	00	00	00	00	00	00	00	Writable<org.apache.
00000006	61	7C	00	00	00	04	00	00	00	01	00	00	01	74	3C	2F<6C
00000007	60	60	6C	76	65	65	72	73	69	67	6E	20	22	33	2E	20	and Verifier+1.0
00000008	22	10	00	65	65	67	69	69	69	67	6E	55	65	6A	65	65	encoding=UTF
00000009	28	22	27	3C	74	72	61	62	61	73	61	65	74	69	67	6E	8177connection
0000000A	00	60	6C	76	72	74	70	70	70	67	6E	60	70	00	00	00	Request+null+M
0000000B	74	74	74	74	74	74	74	74	74	74	74	74	74	74	74	74	top://localhost/
0000000C	2E	61	62	72	72	72	65	70	67	72	74	22	2E	3C	74	72	<and request>
0000000D	60	73	61	67	69	69	67	6E	32	65	70	70	67	72	74	72	Capabilities
0000000E	3C	72	70	67	74	2E	3C	72	67	65	70	67	72	74	65	74	Capabilities
0000000F	60	62	65	72	74	39	39	39	39	39	3C	2F	72	72	65	70	@@@/sape
00000010	72	74	00	00	02	65	72	74	2E	3C	3C	62	72	65	61	74	UnSub+create
00000011	64	44	44	65	74	65	74	69	69	69	3C	3C	3C	3C	3C	3C	date=2012-01-10
00000012	16	1D	31	35	34	31	3A	3A	3A	22	39	3A	35	32	28	31	6-10T11:21:52.10
00000013	74	20	3C	2E	17	65	67	69	69	67	6E	74	65	6A	61	74	65
00000014	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	69	time=processdate
00000015	61	74	65	74	69	65	65	65	3C	3C	30	31	35	32	30	36	date=2012-06-
00000016	31	35	34	31	3A	32	3A	32	3A	22	39	3A	35	32	28	31	10T11:21:52.106
00000017	30	3C	2F	70	67	65	65	65	70	73	65	64	64	61	74	65	@/processdate
00000018	74	69	65	65	72	3C	72	65	70	67	72	74	50	74	61	74	time=processDate
00000019	75	73	49	67	64	65	2E	61	65	74	69	36	36	36	36	36	url=dochecho/c
0000001A	60	67	72	74	50	74	61	74	75	73	70	49	67	64	65	3C	agentStatusCode
0000001B	3C	2F	72	65	65	67	74	74	3C	3C	2F	74	74	72	61	65	</agent>/success
0000001C	61	67	74	69	6E	65	65	65	67	6E	74	69	3C	2F	70	74	<and request>
0000001D	72	61	6E	73	61	63	64	69	67	6E	32	65	70	67	72	74	success+request
0000001E	73	3E															>

Sequence File opened with a hex editor

if you write to HDFS, you can display it on HDFS shell with:

```
1  
2 $ hdfs dfs -text path/report.seq  
3
```

02: Q7 – Q15 Hadoop overview & architecture interview Q&As

02: Convert XML file To Sequence File with Apache Spark – writing & reading

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](#)