»

Text Files

Spark SQL provides `spark.read().text("file_name")` to read a file or directory of text files into a Spark DataFrame, and `dataframe.write().text("path")` to write to a text file. When reading a text file, each line becomes each row that has string “value” column by default. The line separator can be changed as shown in the example below. The `option()` function can be used to customize the behavior of reading or writing, such as controlling behavior of the line separator, compression, and so on.

Scala Java Python

```java
import org.apache.spark.sql.Dataset;
import org.apache.spark.sql.Row;

// A text dataset is pointed to by path.
// The path can be either a single text file or a directory of text files
String path = "examples/src/main/resources/people.txt";

Dataset<Row> df1 = spark.read().text(path);
df1.show();
// +-----------+
// |      value|
// +-----------+
// |Michael, 29|
// |   Andy, 30|
// | Justin, 19|
// +-----------+

// You can use 'lineSep' option to define the line separator.
// The line separator handles all `\r`, `\r\n` and `\n` by default.
Dataset<Row> df2 = spark.read().option("lineSep", ",").text(path);
df2.show();
// +-----------+
// |      value|
// +-----------+
// |    Michael|
// |   29\nAndy|
// | 30\nJustin|
// |       19\n|
// +-----------+

// You can also use 'wholetext' option to read each input file as a single row.
Dataset<Row> df3 = spark.read().option("wholetext", "true").text(path);
df3.show();
//  +--------------------+
//  |               value|
//  +--------------------+
//  |Michael, 29\nAndy...|
//  +--------------------+

// "output" is a folder which contains multiple text files and a _SUCCESS file.
df1.write().text("output");

// You can specify the compression format using the 'compression' option.
df1.write().option("compression", "gzip").text("output_compressed");
```

Find full example code at "examples/src/main/java/org/apache/spark/examples/sql/JavaSQLDataSourceExample.java" in the Spark repo.

Data Source Option

Data source options of text can be set via:

- the `.option`/`.options` methods of
  - `DataFrameReader`
  - `DataFrameWriter`
  - `DataStreamReader`
  - `DataStreamWriter`
- `OPTIONS` clause at CREATE TABLE USING DATA_SOURCE

Property Name	**Default**	**Meaning**	**Scope**

`wholetext`	`false`	If true, read each file from input path(s) as a single row.	read
`lineSep`	`\r`, `\r\n`, `\n` (for reading), `\n` (for writing)	Defines the line separator that should be used for reading or writing.	read/write
`compression`	(none)	Compression codec to use when saving to file. This can be one of the known case-insensitive shorten names (none, bzip2, gzip, lz4, snappy and deflate).	write

Other generic options can be found in Generic File Source Options.

- Parquet Files
- ORC Files
- JSON Files
- CSV Files
- Text Files
- Hive Tables
- JDBC To Other Databases
- Avro Files
- Whole Binary Files
- Troubleshooting

Performance Tuning
Distributed SQL Engine
PySpark Usage Guide for Pandas with Apache Arrow
Migration Guide
SQL Reference