HadoopDfsReadWriteExample

Simple Example to Read and Write files from Hadoop DFS

Reading from and writing to Hadoop DFS is no different from how it is done with other file systems. The example HadoopDFSFileReadWrite.java reads a file from HDFS and writes it to another file on HDFS (copy command).

Hadoop FileSystem API describes the methods available to user. Let us walk through the code to understand how it is done.

Before we Begin

Any Java program can talk to HDFS, provided that program is set up right

1. The classpath contains the Hadoop JAR files and its client-side dependencies. (we are being vague here as those dependencies varies from version to version).
2. The hadoop configuration files on the classpath
3. Log4J on the classpath along with a **log4.properties** resource, or commons-logging preconfigured to use a different logging framework.

Step By Step

Create a FileSystem instance by passing a new Configuration object. Please note that the following example code assumes that the Configuration object will automatically load the **hadoop-default.xml** and **hadoop-site.xml** configuration files. You may need to explicitly add these resource paths if you are not running inside of the Hadoop runtime environment.

```
Configuration conf = new Configuration();
FileSystem fs = FileSystem.get(conf);
```

Given an input/output file name as string, we construct inFile/outFile Path objects. Most of the FileSystem APIs accepts Path objects.

```
Path inFile = new Path(argv[0]);
Path outFile = new Path(argv[1]);
```

Validate the input/output paths before reading/writing.

```
if (!fs.exists(inFile))
  printAndExit("Input file not found");
if (!fs.isFile(inFile))
  printAndExit("Input should be a file");
if (fs.exists(outFile))
  printAndExit("Output already exists");
```

Open inFile for reading.

```
FSDataInputStream in = fs.open(inFile);
```

Open outFile for writing.

```
FSDataOutputStream out = fs.create(outFile);
```

Read from input stream and write to output stream until EOF.

```
while ((bytesRead = in.read(buffer)) > 0) {
  out.write(buffer, 0, bytesRead);
}
```

Close the streams when done.

```
in.close();
out.close();
```