1. Overview

Hadoop provides mainly two classes **FSDataInputStream** for reading a file from HDFS and **FSDataOutputStream** for writing a file to HDFS.

2. Initialize Configuration

First step in communication with HDFS is to initialize Configuration class and set fs.defaultFS property.

```
Configuration configuration = new Configuration();
configuration.set("fs.defaultFS", "hdfs://localhost:9000");
```

3. Create Directory in HDFS

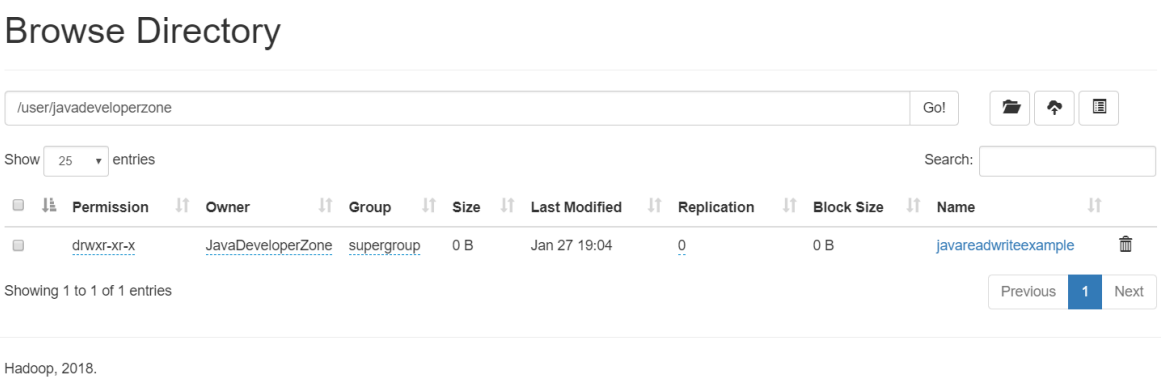
Hadoop FileSystem class provide all the admin related functionality like create file or directory, delete file etc. mkDirs method is used to create a directory under HDFS.

Example

```
public static void createDirectory() throws IOException {
Configuration configuration = new Configuration();
configuration.set("fs.defaultFS", "hdfs://localhost:9000");
FileSystem fileSystem = FileSystem.get(configuration);
String directoryName = "javadeveloperzone/javareadwriteexample";
Path path = new Path(directoryName);
fileSystem.mkdirs(path);
}
```

Output

Go to HDFS web view and everything is running fine you will see a directory javareadwriteexample under /user/javadeveloperzone path.



4. Write File to HDFS

FSDataOutputStream class used to write data to HDFS file. It also provides various methods like writeUTF, writeInt, WriteChar etc..Here we have wrapped FSDataOutputStream to BufferedWrite class.

```
public static void writeFileToHDFS() throws IOException {
Configuration configuration = new Configuration();
configuration.set("fs.defaultFS", "hdfs://localhost:9000");
FileSystem fileSystem = FileSystem.get(configuration);
//Create a path
String fileName = "read_write_hdfs_example.txt";
Path hdfsWritePath = new Path("/user/javadeveloperzone/javareadwriteexample/" + fileName);
FSDataOutputStream fsDataOutputStream = fileSystem.create(hdfsWritePath,true);
BufferedWriter bufferedWriter = new BufferedWriter(new
OutputStreamWriter(fsDataOutputStream,StandardCharsets.UTF_8));
bufferedWriter.write("Java API to write data in HDFS");
bufferedWriter.newLine();
bufferedWriter.close();
fileSystem.close();
}
```

Append Data to File

FileSystem class append method is used to append data to an existing file.

```
public static void appendToHDFSFile() throws IOException {
Configuration configuration = new Configuration();
configuration.set("fs.defaultFS", "hdfs://localhost:9000");
FileSystem fileSystem = FileSystem.get(configuration);
//Create a path
String fileName = "read_write_hdfs_example.txt";
Path hdfsWritePath = new Path("/user/javadeveloperzone/javareadwriteexample/" + fileName);
FSDataOutputStream fsDataOutputStream = fileSystem.append(hdfsWritePath);
BufferedWriter bufferedWriter = new BufferedWriter(new
OutputStreamWriter(fsDataOutputStream,StandardCharsets.UTF_8));
bufferedWriter.write("Java API to append data in HDFS file");
bufferedWriter.newLine();
bufferedWriter.close();
fileSystem.close();
}
```

Read File From HDFS

FSDataInputStream class provide facility to read a file from HDFS.

Example

```
public static void readFileFromHDFS() throws IOException {
Configuration configuration = new Configuration();
configuration.set("fs.defaultFS", "hdfs://localhost:9000");
FileSystem fileSystem = FileSystem.get(configuration);
//Create a path
String fileName = "read_write_hdfs_example.txt";
Path hdfsReadPath = new Path("/user/javadeveloperzone/javareadwriteexample/" + fileName);
//Init input stream
FSDataInputStream inputStream = fileSystem.open(hdfsReadPath);
//Classical input stream usage
String out= IOUtils.toString(inputStream, "UTF-8");
System.out.println(out);
/*BufferedReader bufferedReader = new BufferedReader(
new InputStreamReader(inputStream, StandardCharsets.UTF_8));
String line = null;
while ((line=bufferedReader.readLine())!=null){
System.out.println(line);
}*/
inputStream.close();
fileSystem.close();
}
```

```
package com.javadeveloperzone;

import org.apache.commons.io.IOUtils;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FSDataInputStream;
import org.apache.hadoop.fs.FSDataOutputStream;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import java.io.*;
import java.nio.charset.StandardCharsets;

public class ReadWriteHDFSExample {

    public static void main(String[] args) throws IOException {
      ReadWriteHDFSExample.readFileFromHDFS();
    }

    public static void readFileFromHDFS() throws IOException {
        Configuration configuration = new Configuration();
        configuration.set("fs.defaultFS", "hdfs://localhost:9000");
        FileSystem fileSystem = FileSystem.get(configuration);
        //Create a path
        String fileName = "read_write_hdfs_example.txt";
        Path hdfsReadPath = new Path("/user/javadeveloperzone/javareadwriteexample/" + fileName);
        //Init input stream
        FSDataInputStream inputStream = fileSystem.open(hdfsReadPath);
        //Classical input stream usage
        String out= IOUtils.toString(inputStream, "UTF-8");
        System.out.println(out);

        /*BufferedReader bufferedReader = new BufferedReader(
         new InputStreamReader(inputStream, StandardCharsets.UTF_8));

        String line = null;
        while ((line=bufferedReader.readLine())!=null){
         System.out.println(line);
        }*/

        inputStream.close();
        fileSystem.close();
    }

    public static void writeFileToHDFS() throws IOException {
        Configuration configuration = new Configuration();
        configuration.set("fs.defaultFS", "hdfs://localhost:9000");
        FileSystem fileSystem = FileSystem.get(configuration);
        //Create a path
        String fileName = "read_write_hdfs_example.txt";
        Path hdfsWritePath = new Path("/user/javadeveloperzone/javareadwriteexample/" +
fileName);
        FSDataOutputStream fsDataOutputStream = fileSystem.create(hdfsWritePath,true);
```

```
        BufferedWriter bufferedWriter = new BufferedWriter(new
OutputStreamWriter(fsDataOutputStream,StandardCharsets.UTF_8));
        bufferedWriter.write("Java API to write data in HDFS");
        bufferedWriter.newLine();
        bufferedWriter.close();
        fileSystem.close();
    }

    public static void appendToHDFSFile() throws IOException {
        Configuration configuration = new Configuration();
        configuration.set("fs.defaultFS", "hdfs://localhost:9000");
        FileSystem fileSystem = FileSystem.get(configuration);
        //Create a path
        String fileName = "read_write_hdfs_example.txt";
        Path hdfsWritePath = new Path("/user/javadeveloperzone/javareadwriteexample/" +
fileName);
        FSDataOutputStream fsDataOutputStream = fileSystem.append(hdfsWritePath);

        BufferedWriter bufferedWriter = new BufferedWriter(new
OutputStreamWriter(fsDataOutputStream,StandardCharsets.UTF_8));
        bufferedWriter.write("Java API to append data in HDFS file");
        bufferedWriter.newLine();
        bufferedWriter.close();
        fileSystem.close();
    }

    public static void createDirectory() throws IOException {
        Configuration configuration = new Configuration();
        configuration.set("fs.defaultFS", "hdfs://localhost:9000");
        FileSystem fileSystem = FileSystem.get(configuration);
        String directoryName = "javadeveloperzone/javareadwriteexample";
        Path path = new Path(directoryName);
        fileSystem.mkdirs(path);
    }

    public static void checkExists() throws IOException {
        Configuration configuration = new Configuration();
        configuration.set("fs.defaultFS", "hdfs://localhost:9000");
        FileSystem fileSystem = FileSystem.get(configuration);
        String directoryName = "javadeveloperzone/javareadwriteexample";
        Path path = new Path(directoryName);
        if(fileSystem.exists(path)){
            System.out.println("File/Folder Exists : "+path.getName());
        }else{
            System.out.println("File/Folder does not Exists : "+path.getName());
        }
    }
}
```