# HDFS读写数据流分析

**【实验名称】**

HDFS读写数据流分析

**【实验目的】**

通过这个实验,学会通过java客户端通过API对HDFS读取数据流程.

**【实验要求】**

要求实验结束时，每位学生均已搭建HDFS开发环境；编写HDFS读写代码，并在机子上执行了该读写程序，了解HDFS读写文件的调用流程，理解HDFS读写文件的原理。

**【实验环境】**

HDFS的基础架构了解

Java基础知识

**【实验步骤】**

## 1.hdfs java客户端操作

### 1.1 创建java maven 项目



### 1.2配置pom.xml文件，加载jar包



### 1.3、创建Hdfs连接



### 1.4、把hdfs上的文件写到linux目录下



### 1.5.1、java代码(读取hdfs)

package hhh;

import java.io.File;

import java.io.FileOutputStream;

import java.io.OutputStream;

import java.net.URI;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.FSDataInputStream;

import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.fs.Path;

public class Demo1 {

 public static void main(String[] args) throws Exception {

 //创建configuration对象

 Configuration conf = new Configuration();

 //创建FileSystem对象

 //需求：查看hdfs集群服务器/user/zyh/passwd.txt的内容

 FileSystem fs = FileSystem.get(new URI("hdfs://172.16.16.131:9000"), conf, "root");

 // args[0] hdfs://1.0.0.3:9000/user/zyh/passwd.txt

 // args[0] file:///etc/passwd.txt

 FSDataInputStream is = fs.open(new Path("hdfs://172.16.16.131:9000/input/wordcount.txt"));

 OutputStream os=new FileOutputStream(new File("/root/b.txt"));

 byte[] buff= new byte[1024];

 int length = 0;

 while ((length=is.read(buff))!=-1){

 System.out.println(new String(buff,0,length));

 os.write(buff,0,length);

 os.flush();

 }

 System.out.println(fs.getClass().getName());

 }

}

### 1.5.2、java代码(写入hdfs)

**package** hhh;

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.net.URI;

**import** org.apache.commons.io.IOUtils;

**import** org.apache.hadoop.conf.Configuration;

**import** org.apache.hadoop.fs.FSDataOutputStream;

**import** org.apache.hadoop.fs.FileSystem;

**import** org.apache.hadoop.fs.Path;

**public** **class** Demo2 {

 **public** **static** **void** main(String[] args) **throws** Exception {

 //创建configuration对象

 Configuration conf = **new** Configuration();

 //创建FileSystem对象

 //需求：查看hdfs集群服务器/user/zyh/passwd.txt的内容

 FileSystem fs = FileSystem.*get*(**new** URI("hdfs://172.16.16.131:9000"), conf, "root");

 // args[0] hdfs://1.0.0.3:9000/user/zyh/passwd.txt

 // args[0] file:///etc/passwd.txt

 FileInputStream inputStream = **new** FileInputStream(**new** File("/root/b.txt"));

 FSDataOutputStream outputStream = fs.create(**new** Path("/input/test3.txt"),**true**);

 IOUtils.*copy*(inputStream,outputStream);

 }

}

### 1.6、pom.xml配置文件

 <<dependencies>

 <dependency>

 <groupId>org.apache.hadoop</groupId>

 <artifactId>hadoop-common</artifactId>

 <version>2.8.1</version>

 </dependency>

 <dependency>

 <groupId>org.apache.hadoop</groupId>

 <artifactId>hadoop-client</artifactId>

 <version>2.8.1</version>

 </dependency>

 <dependency>

 <groupId>org.apache.hadoop</groupId>

 <artifactId>hadoop-hdfs</artifactId>

 <version>2.8.1</version>

 </dependency>

 <dependency>

 <groupId>org.apache.hadoop</groupId>

 <artifactId>hadoop-mapreduce-client-core</artifactId>

 <version>2.8.1</version>

 </dependency>

 <dependency>

 <groupId>org.apache.hadoop</groupId>

 <artifactId>hadoop-auth</artifactId>

 <version>2.8.1</version>

 </dependency>

 <dependency>

 <groupId>log4j</groupId>

 <artifactId>log4j</artifactId>

 <version>1.2.17</version>

 </dependency>

 <dependency>

 <groupId>commons-logging</groupId>

 <artifactId>commons-logging</artifactId>

 <version>1.2</version>

 </dependency>

 <dependency>

 <groupId>com.google.guava</groupId>

 <artifactId>guava</artifactId>

 <version>19.0</version>

 </dependency>

 <dependency>

 <groupId>commons-collections</groupId>

 <artifactId>commons-collections</artifactId>

 <version>3.2.2</version>

 </dependency>

 <dependency>

 <groupId>commons-cli</groupId>

 <artifactId>commons-cli</artifactId>

 <version>1.2</version>

 </dependency>

 <dependency>

 <groupId>commons-lang</groupId>

 <artifactId>commons-lang</artifactId>

 <version>2.6</version>

 </dependency>

 <dependency>

 <groupId>commons-configuration</groupId>

 <artifactId>commons-configuration</artifactId>

 <version>1.9</version>

 </dependency>

 <dependency>

 <groupId>org.apache.avro</groupId>

 <artifactId>avro</artifactId>

 <version>1.7.7</version>

 </dependency>

 <dependency>

 <groupId>commons-io</groupId>

 <artifactId>commons-io</artifactId>

 <version>2.5</version>

 </dependency>

 </dependencies>

### 1.7 把java打包成jar

#### 1.7.1 选中项目右键，选择Export,选中JAR file,然后next下一步



#### 1.7.2选择要打包的项目和地址,然后next下一步



#### 1.7.3 选择打包的class,选择OK 然后Finish



## 2.Hadoop运行打包好的jar文件

### 2.1 将打包好的jar上传到linux

#### 2.1.1 右键linux虚拟机，选择Connect SFTP Session 进入sftp



#### 2.1.2将jar直接拖到sftp上



#### 2.1.3 然后执行下面命令

[root@hadoop1]# hadoop jar WordCountApp.jar



#### 2.1.4 执行读取成功，查看文件数据

[root@hadoop1]# vi b.txt



读取HDFS数据操作成功

#### 2.1.5 执行写入成功，查看文件数据



写入HDFS数据操作成功

至此，本节实验结束！