|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 姓名：崔文帅 | | 学号：2025060163 | 专业年级：  软工2020级 | 班级：14955017 |
| 分组：第1组 | | 实验室：计算机大楼402机房 | 指导教师：何洪辉 | 实验日期：2023-03-08 |
| **实验的准备阶段** | **课程名称** | **大数据基础** | | |
| **实验名称** | HDFS API | | |
| **实验目的** | 1. 熟练HDFS-API 2. 掌握Java编程访问HDFS | | |
| **实验内容** | 1. Java编程实现HDFS下创建目录 2. Java编程实现HDFS下显示目录列表 3. Java编程实现HDFS下删除目录 4. Java编程实现向HDFS上传文件 5. Java编程实现从HDFS下载文件 | | |
| **实验类型**  （打☑） | ☑验证性 □演示性□设计性□综合性 | | |
| **实验的重点、难点** | 重点：HDFS-API  难点：编程实现 | | |
| **实验环境** | centos6.5 + jdk1.7.0\_79 + hadoop2.4.1 (注意：也可以Windows环境） | | |
| **实验的实施阶段** | **实验步骤及实验结果** | **一、启动Hadoop服务**         1.1查看服务状态。在命令终端，执行命令“jps”（查看hadoop服务是否已经启动，如未启动，启动服务）。            1.2 启动hadoop服务。启动hadoop服务可以通过一次性启动hadoop所有服务，执行命令“start-all.sh”，。还可以通过单独启动    **二、 创建java项目包**          2.1 打开ecplise开发工具，点击File选择“New”->“Java Project”新建名为“Hadoop”的java项目，点击“Hadoop”项目右键，选择“New” –>“Package”；如图3所示            2.2 输入包名称“com.hdfs”点击完成。            2.3 新建java 类。选中包名并右键点击，选择“New”->“Class”。            2.4 在name 项输入“ HdfsTest”类名称，点击Finish完成。            2.5 在创建的项目目录下右击选择“New”—“Folder”创建文件夹libs。          2.6 拷贝桌面lib文件夹下的所有hadoop jar包到libs文件夹下。          2.7 选中libs下的所有jar包，右键点击，然后选择“Build Path”下的“add to build path”即可把所有jar包添加到path环境中。 **三、编写程序** package com.hdfs;  import java.net.URI; import org.apache.hadoop.conf.Configuration; import org.apache.hadoop.fs.FileSystem; import org.apache.hadoop.fs.Path; import org.junit.Before; import org.junit.Test;   public class HdfsTest {     // 获取HADOOP FileSystem对象     private FileSystem fs = null;     /\*\*      \* 初始化环境变量      \*/     @Before     public void init() throws Exception {        /\*         \* new URI("hdfs://192.168.0.131:9000"):连接HADOOP连接 RUL         \* new Configuration():使用HADOOP 默认配置         \* "root"：登录用户         \*/        fs = FileSystem.get(new URI("hdfs://192.168.1.2:9000"),new Configuration(), "root");     }     /\*\*      \* 创建目录      \*/     @Test     public void testMkdir() throws Exception {        boolean flag = fs.mkdirs(new Path("/javaApi/mk/dir1/dir2"));        System.out.println(flag  ? "创建成功" : "创建失败");     } } **四、程序运行**         4.1 在项目栏目最左侧，选中要运行的方法“testMkdir”—>点击右键—>选择“run as”—>选择“Junit Test”，等待执行结果。            4.2 显示运行结果 。    **五、程序结果验证**          5.1 通过执行hadoop shell 命令“hadoop fs –ls –R /javaApi”进行验证。    **一、启动Hadoop服务**          1.1查看服务状态。在命令终端，执行命令“jps”（查看hadoop服务是否已经启动，如未启动，启动服务）。如图1所示            1.2 启动hadoop服务。启动hadoop服务可以通过一次性启动hadoop所有服务，执行命令“start-all.sh”，如图2所示。还可以通过单独启动    图2  **二、创建java项目包**          2.1 打开ecplise开发工具，点击File选择“New”->“Java Project”新建名为“Hadoop”的java项目，点击“Hadoop”项目右键，选择“New” –>“Package”；如图3所示    图3      2.2 输入名称“com.hdfs”点击完成。如图4所示    图4          2.3 新建java 类。选中包名并右键点击，选择“New”->“Class”。如图5所示    图5          2.4 在name 项输入“ HdfsTest”类名称，点击Finish完成。如图6所示    图6          2.5 在创建的项目目录下右击选择“New”—“Folder”创建文件夹libs。          2.6 拷贝桌面lib文件夹下的所有hadoop jar包到libs文件夹下。          2.7 选中libs下的所有jar包，右键点击，然后选择“Build Path”下的“add to build path”即可把所有jar包添加到path环境中。    **三、编写程序**   |  | | --- | | **package** com.hdfs;  **import**  java.net.URI;  **import**  java.util.ArrayList;  **import**  java.util.List;  **import**  org.apache.hadoop.conf.Configuration;  **import**  org.apache.hadoop.fs.FileStatus;  **import**  org.apache.hadoop.fs.FileSystem;  **import** org.apache.hadoop.fs.Path;  **import**  org.junit.After;  **import**  org.junit.Before;  **import**  org.junit.Test;  **public** **class** HdfsTest {      // 获取HADOOP FileSystem对象  **private**  FileSystem fs = **null**;  **private**  List<String> hdfsPathsLists;      /\*\*       \* 初始化环境变量       \*/      @Before  **public** **void** init() **throws** Exception  {         /\*          \* new URI("hdfs://192.168.0.131:9000"):连接HADOOP连接 RUL new          \* Configuration():使用HADOOP 默认配置          \* "root"：登录用户          \*/         fs = FileSystem.*get*(**new** URI("hdfs://192.168.1.2:9000"),**new** Configuration(), "root");      }      @Test  **public** **void** getDirList() **throws**  Exception{         //初始化存放目录的List         hdfsPathsLists = **new**  ArrayList<>();         //调用获取目录方法         getHdfsPaths(**new** Path("/"));         //循环输入结果  **for**(String p:hdfsPathsLists){             System.*out*.println(p);         }      }      /\*\*       \* 递归方法便利获取目录及目录下文件       \* **@param** path 要获取的目录       \* **@throws** Exception 异常       \*/  **private** **void**  getHdfsPaths(Path path) **throws** Exception{         FileStatus[]  dirs = fs.listStatus(path);  **for** (FileStatus s :  dirs){                hdfsPathsLists.add(s.getPath().toString());  **if**(s.isDirectory()){                        getHdfsPaths(s.getPath());                    }             }         }      @After  **public** **void** close()**throws**  Exception{         fs.close();      }  } |   **四、程序运行**          4.1 在项目栏目最左侧，选中要运行的方法“getDirList”—>点击右键—>选择“run as”—>选择“Junit Test”，等待执行结果。如图7所示    图7           4.2 显示运行结果。如图8所示      图8  **五、程序结果验证**          5.1 通过hadoop shell 命令进行验证。如图9所示    **一、启动Hadoop服务**         1.1 查看服务状态。在命令终端，执行命令“jps”（查看hadoop服务是否已经启动，如未启动，启动服务）。如图1所示    图1          1.2 启动hadoop服务。启动hadoop服务可以通过一次性启动hadoop所有服务，执行命令“start-all.sh”，如图2所示。还可以通过单独启动    图2          1.3 通过hdfs dfs -touchz /java在HDFS上创建文件。如图3所示。    图3  **二、 创建java项目包**          2.1 打开ecplise开发工具，点击File选择“New”->“Java Project”新建名为“Hadoop”的java项目，点击“Hadoop”项目右键，选择“New” –>“Package”；如图4所示    图4          2.2 输入名称“com.hdfs”点击完成。如图5所示    图5          2.3 新建java 类。选中包名并右键点击，选择“New”->“Class”。如图6所示    图6          2.4 在name 项输入“ HdfsTest”类名称，点击Finish完成。如图7所示    图7          2.5 在创建的项目目录下右击选择“New”—“Folder”创建文件夹libs。          2.6 拷贝桌面lib文件夹下的所有hadoop jar包到libs文件夹下。          2.7 选中libs下的所有jar包，右键点击，然后选择“Build Path”下的“add to build path”即可把所有jar包添加到path环境中。    **三、编写程序** package com.hdfs;  import  java.net.URI; import  org.apache.hadoop.conf.Configuration; import  org.apache.hadoop.fs.FileSystem; import  org.apache.hadoop.fs.Path; import  org.junit.Before; import  org.junit.Test;  public class HdfsTest {     // 获取HADOOP FileSystem对象     private  FileSystem fs = null;     /\*\*      \* 初始化环境变量      \*/     @Before     public void init() throws Exception  {        /\*         \* new URI("hdfs://192.168.0.131:9000"):连接HADOOP连接 RUL new         \* Configuration():使用HADOOP 默认配置        \* "root"：登录用户         \*/        fs = FileSystem.get(new URI("hdfs://192.168.1.2:9000"),new Configuration(), "root");     }     /\*\*      \* 删除目录      \*/     @Test     public void testRMdir() throws  Exception  {     boolean flag = fs.deleteOnExit(new Path("/java "));     System.out.println(flag ? "删除成功" : "删除失败");     } } **四、程序运行**         4.1 在项目栏目最左侧，选中要运行的方法“testRMdir”—>点击右键—>选择“run as”—>选择“Junit Test”，等待执行结果。如图8所示    图8          4.2 显示运行结果。 如图9所示    图9  **五、程序结果验证**          5.1 通过hadoop shell 命令进行验证,如图10所示    **【实验目的】**      1）熟悉HDFS java编程—上传文件  **【实验原理】**      通过hadoop 提供的java api对hdfs进行操作。  **【实验环境】**      本次环境是：centos6.5 + jdk1.7 + hadoop2.4.1(64位)+ecplise  **【实验步骤】**  **一、启动Hadoop服务**          1.1查看服务状态。在命令终端，执行命令“jps”（查看hadoop服务是否已经启动，如未启动，启动服务）。如图1所示    图1          1.2 启动hadoop服务。启动hadoop服务可以通过一次性启动hadoop所有服务，执行命令“start-all.sh”，如图2所示。还可以通过单独启动    图2          1.3 通过touch /simple/HelloWorld.txt在simple目录下创建HelloWorld.txt文件。如图3所示    图3  **二、创建java项目包**          2.1 打开ecplise开发工具，点击File选择“New”->“Java Project”新建名为“Hadoop”的java项目，点击“Hadoop”项目右键，选择“New” –>“Package”；如图4所示    图4          2.2 输入名称“com.hdfs”点击完成。如图5所示    图5          2.3 新建java 类。选中包名并右键点击，选择“New”->“Class”。如图6所示    图6          2.4 在name 项输入“ HdfsTest”类名称，点击Finish完成。如图7所示    图7          2.5 在创建的项目目录下右击选择“New”—“Folder”创建文件夹libs。          2.6 拷贝桌面lib文件夹下的所有hadoop jar包到libs文件夹下。          2.7 选中libs下的所有jar包，右键点击，然后选择“Build Path”下的“add to build path”即可把所有jar包添加到path环境中。  **三、编写程序**   |  | | --- | | package com.hdfs;  import java.net.URI;  import java.util.ArrayList;  import java.util.List;  import org.apache.hadoop.conf.Configuration;  import org.apache.hadoop.fs.FileStatus;  import org.apache.hadoop.fs.FileSystem;  import org.apache.hadoop.fs.Path;  import org.junit.After;  import org.junit.Before;  import org.junit.Test;  public class HdfsTest {      //  获取HADOOP  FileSystem对象      private  FileSystem fs = null;      private  List<String> hdfsPathsLists;      /\*\*       \* 初始化环境变量       \*/      @Before      public  void init() throws Exception {         /\*          \* new  URI("hdfs://192.168.0.201:9000"):连接HADOOP连接 RUL new          \* Configuration():使用HADOOP 默认配置          \* "root"：登录用户          \*/         Configuration  con = new Configuration();         con.setBoolean("dfs.support.append",  true);         fs  = FileSystem.get(new URI("hdfs://192.168.1.2:9000"),con,  "root");      }      /\*\*       \* 将本地文件上传到HDFS上       \*/      @Test      public  void testUploadFile()throws Exception {         String src="/simple/HelloWorld.txt ";         String  hdfsDst= "/javaApi";         fs.copyFromLocalFile(new  Path(src), new Path(hdfsDst));          System.out.println("upload  sucess");      }  } |   **四、程序运行**          4.1 在项目栏目最左侧，选中要运行的方法“testUploadFile”—>点击右键—>选择“run as”—>选择“Junit Test”，等待执行结果。如图8所示    图8           4.2 显示运行结果。如图9所示    图9  **五、程序结果验证**           5.1 通过hadoop shell 命令进行验证 。如图10所示    图10  **【实验目的】**      1）熟悉HDFS java编程—下载文件  **【实验原理】**      通过hadoop 提供的java api对hdfs进行操作。  **【实验环境】**      本次环境是：centos6.5 + jdk1.7 + hadoop2.4.1(64位)+ecplise  **【实验步骤】**  **一、启动Hadoop服务**          1.1 查看服务状态。在命令终端，执行命令“jps”（查看hadoop服务是否已经启动，如未启动，启动服务）。如图1所示    图1          1.2 启动hadoop服务。启动hadoop服务可以通过一次性启动hadoop所有服务，执行命令“start-all.sh”，如图2所示。还可以通过单独启动    图2          1.3 通过hdfs dfs -touchz /java在HDFS上创建文件。如图3所示。    图3  **二、创建java项目包**          2.1 打开ecplise开发工具，点击File选择“New”->“Java Project”新建名为“Hadoop”的java项目，点击“Hadoop”项目右键，选择“New” –>“Package”；如图4所示    图4          2.2 输入名称“com.hdfs”点击完成。如图5所示    图5          2.3 新建java 类。选中包名并右键点击，选择“New”->“Class”。如图6所示    图6          2.4 在name 项输入“ HdfsTest”类名称，点击Finish完成。如图7所示    图7          2.5 在创建的项目目录下右击选择“New”—“Folder”创建文件夹libs。          2.6 拷贝桌面lib文件夹下的所有hadoop jar包到libs文件夹下。          2.7 选中libs下的所有jar包，右键点击，然后选择“Build Path”下的“add to build path”即可把所有jar包添加到path环境中。。  **三、编写程序**   |  | | --- | | **package** com.hdfs;  **import** java.io.IOException;  **import**  java.net.URI;  **import**  org.apache.hadoop.conf.Configuration;  **import**  org.apache.hadoop.fs.FileSystem;  **import** org.apache.hadoop.fs.Path;  **import**  org.junit.Before;  **import**  org.junit.Test;  **public** **class** HdfsTest {      // 获取HADOOP FileSystem对象  **private**  FileSystem fs = **null**;      /\*\*       \* 初始化环境变量       \*/      @Before  **public** **void** init() **throws** Exception  {         /\*          \* new URI("hdfs://192.168.0.131:9000"):连接HADOOP连接 RUL new          \* Configuration():使用HADOOP 默认配置          \* "root"：登录用户          \*/         fs = FileSystem.*get*(**new** URI("hdfs://192.168.1.2:9000"),**new** Configuration(), "root");      }      /\*\*       \* 文件下载到本地       \*/  @Test  **public** **void**  testDownloadFile() **throws** Exception{         String  src= "/java ";         String  hdfsDst="/root";         fs.copyToLocalFile(**new** Path(src), **new** Path(hdfsDst));  System.out.println("downLoad  sucess");      }  } |   **四、程序运行**          4.1 在项目栏目最左侧，选中要运行的方法“testDownloadFile”—>点击右键—>选择“run as”—>选择“Junit Test”，等待执行结果。如图8所示    图8           4.2 显示运行结果。如图9所示    图9  **五、程序结果验证**           5.1 通过hadoop shell 命令进行验证。如图10所示    图10 | | |
| **实验结果的处理阶段** | **实验结果的分析与总结** |  | | |