**总结**

# Map

## 遍历Map

1. Map.keySet()

|  |
| --- |
| Set**<**Map**.**Entry**<**String**,** String**>>** entrySet **=** map**.**entrySet**();**  //将关系集合entrySet进行迭代，存放到迭代器中  Iterator**<**Map**.**Entry**<**String**,** String**>>** it2 **=** entrySet**.**iterator**();**  **while** **(**it2**.**hasNext**())** **{**  //获取Map.Entry关系对象me  Map**.**Entry**<**String**,** String**>** me **=** it2**.**next**();**  //通过关系对象获取key  String key2 **=** me**.**getKey**();**  //通过关系对象获取value  String value2 **=** me**.**getValue**();**  System**.**out**.**println**(**"key: " **+** key2 **+** "-->value: " **+** value2**);**  **}** |

1. Map.entrySet()

|  |
| --- |
| 通过entrySet**()**方法将map集合中的映射关系取出（这个关系就是Map**.**Entry类型）  Set**<**Map**.**Entry**<**String**,** String**>>** entrySet **=** map**.**entrySet**();**  //将关系集合entrySet进行迭代，存放到迭代器中  Iterator**<**Map**.**Entry**<**String**,** String**>>** it2 **=** entrySet**.**iterator**();**  **while** **(**it2**.**hasNext**())** **{**  //获取Map.Entry关系对象me  Map**.**Entry**<**String**,** String**>** me **=** it2**.**next**();**  //通过关系对象获取key  String key2 **=** me**.**getKey**();**  //通过关系对象获取value  String value2 **=** me**.**getValue**();**  System**.**out**.**println**(**"key: " **+** key2 **+** "-->value: " **+** value2**);**  **}** |

# Date

格式转变：java.Utile.Date –> String

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");--时间格式

Date date = new Date();

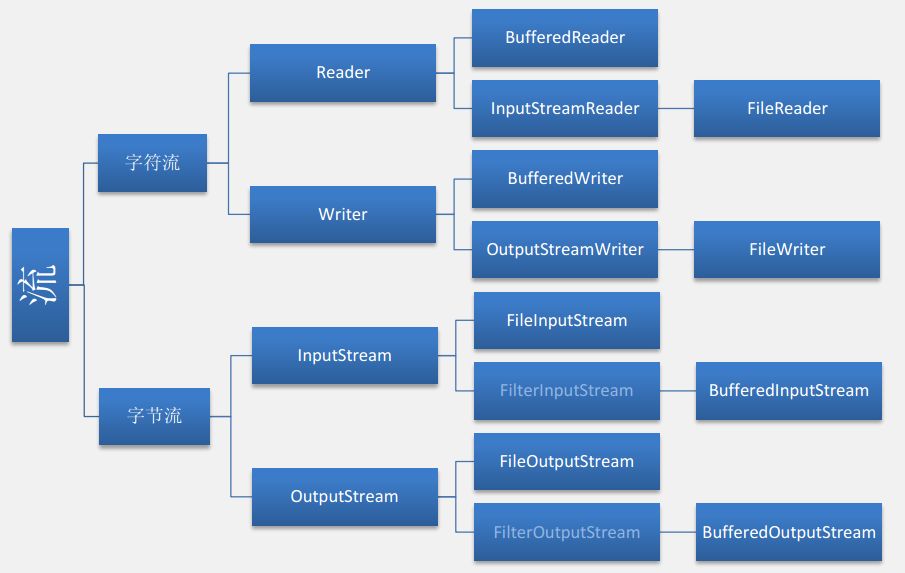
String time = sdf.format(date);

格式转变： String –> java.Utile.Date

String str = "2007-1-18";

Date date = sdf.parse(str);

# I/O操作



## 1创建文件夹

//import java.io.\*;

File myFolderPath = new File(str1);

try {

if (!myFolderPath.exists()) {

myFolderPath.mkdir();

}

}

catch (Exception e) {

System.out.println("新建目录操作出错");

e.printStackTrace();

}

## 2.创建文件

//import java.io.\*;

File myFilePath = new File(str1);

try {

if (!myFilePath.exists()) {

myFilePath.createNewFile();

}

FileWriter resultFile = new FileWriter(myFilePath);

PrintWriter myFile = new PrintWriter(resultFile);

myFile.println(str2);

resultFile.close();

}

catch (Exception e) {

System.out.println("新建文件操作出错");

e.printStackTrace();

}

## 3.删除文件

//import java.io.\*;

File myDelFile = new File(str1);

try {

myDelFile.delete();

}

catch (Exception e) {

System.out.println("删除文件操作出错");

e.printStackTrace();

}

## 4.删除文件夹

//import java.io.\*;

File delFolderPath = new File(str1);

try {

delFolderPath.delete(); //删除空文件夹

}

catch (Exception e) {

System.out.println("删除文件夹操作出错");

e.printStackTrace();

}

## 5.删除一个文件下夹所有的文件夹

//import java.io.\*;

File delfile=new File(str1);

File[] files=delfile.listFiles();

for(int i=0;i<files.length;i++){

if(files[i].isDirectory()){

files[i].delete();

}

}

## 6.清空文件夹

//import java.io.\*;

File delfilefolder=new File(str1);

try {

if (!delfilefolder.exists()) {

delfilefolder.delete();

}

delfilefolder.mkdir();

}

catch (Exception e) {

System.out.println("清空目录操作出错");

e.printStackTrace();

}

## 7.读取文件

//import java.io.\*;

// 逐行读取数据

FileReader fr = new FileReader(str1);

BufferedReader br = new BufferedReader(fr);

String str2 = br.readLine();

while (str2 != null) {

str3

str2 = br.readLine();

}

br.close();

fr.close();

## 8.写入文件

//import java.io.\*;

// 将数据写入文件

try {

FileWriter fw = new FileWriter(str1);

fw.write(str2);

fw.flush();

fw.close();

} catch (IOException e) {

e.printStackTrace();

}

## 9.写入随机文件

//import java.io.\*;

try {

RandomAcessFile logFile=new RandomAcessFile(str1,"rw");

long lg=logFile.length();

logFile.seek(str2);

logFile.writeByte(str3);

}catch(IOException ioe){

System.out.println("无法写入文件："+ioe.getMessage());

}

## 10.读取文件属性

//import java.io.\*;

// 文件属性的取得

File f = new File(str1);

if (af.exists()) {

System.out.println(f.getName() + "的属性如下： 文件长度为：" + f.length());

System.out.println(f.isFile() ? "是文件" : "不是文件");

System.out.println(f.isDirectory() ? "是目录" : "不是目录");

System.out.println(f.canRead() ? "可读取" : "不");

System.out.println(f.canWrite() ? "是隐藏文件" : "");

System.out.println("文件夹的最后修改日期为：" + new Date(f.lastModified()));

} else {

System.out.println(f.getName() + "的属性如下：");

System.out.println(f.isFile() ? "是文件" : "不是文件");

System.out.println(f.isDirectory() ? "是目录" : "不是目录");

System.out.println(f.canRead() ? "可读取" : "不");

System.out.println(f.canWrite() ? "是隐藏文件" : "");

System.out.println("文件的最后修改日期为：" + new Date(f.lastModified()));

}

if(f.canRead()){

str2

}

if(f.canWrite()){

str3

}

## 11.写入属性

//import java.io.\*;

File filereadonly=new File(str1);

try {

boolean b=filereadonly.setReadOnly();

}

catch (Exception e) {

System.out.println("拒绝写访问："+e.printStackTrace());

}

## 12.枚举一个文件夹中的所有文件

//import java.io.\*;

//import java.util.\*;

LinkedList<String> folderList = new LinkedList<String>();

folderList.add(str1);

while (folderList.size() > 0) {

File file = new File(folderList.peek());

folderList.removeLast();

File[] files = file.listFiles();

ArrayList<File> fileList = new ArrayList<File>();

for (int i = 0; i < files.length; i++) {

if (files[i].isDirectory()) {

folderList.add(files[i].getPath());

} else {

fileList.add(files[i]);

}

}

for (File f : fileList) {

str2=f.getAbsoluteFile();

str3

}

}

## 13.复制文件夹

//import java.io.\*;

//import java.util.\*;

LinkedList<String> folderList = new LinkedList<String>();

folderList.add(str1);

LinkedList<String> folderList2 = new LinkedList<String>();

folderList2.add(str2+ str1.substring(str1.lastIndexOf("\\")));

while (folderList.size() > 0) {

(new File(folderList2.peek())).mkdirs(); // 如果文件夹不存在 则建立新文件夹

File folders = new File(folderList.peek());

String[] file = folders.list();

File temp = null;

try {

for (int i = 0; i < file.length; i++) {

if (folderList.peek().endsWith(File.separator)) {

temp = new File(folderList.peek() + File.separator

+ file[i]);

} else {

temp = new File(folderList.peek() + File.separator + file[i]);

}

if (temp.isFile()) {

FileInputStream input = new FileInputStream(temp);

FileOutputStream output = new FileOutputStream(

folderList2.peek() + File.separator + (temp.getName()).toString());

byte[] b = new byte[5120];

int len;

while ((len = input.read(b)) != -1) {

output.write(b, 0, len);

}

output.flush();

output.close();

input.close();

}

if (temp.isDirectory()) {// 如果是子文件夹

for (File f : temp.listFiles()) {

if (f.isDirectory()) {

folderList.add(f.getPath());

folderList2.add(folderList2.peek()

+ File.separator + f.getName());

}

}

}

}

} catch (Exception e) {

//System.out.println("复制整个文件夹内容操作出错");

e.printStackTrace();

}

folderList.removeFirst();

folderList2.removeFirst();

}

## 14.复制一个文件夹下所有的文件夹到另一个文件夹下

//import java.io.\*;

//import java.util.\*;

File copyfolders=new File(str1);

File[] copyfoldersList=copyfolders.listFiles();

for(int k=0;k<copyfoldersList.length;k++){

if(copyfoldersList[k].isDirectory()){

ArrayList<String>folderList=new ArrayList<String>();

folderList.add(copyfoldersList[k].getPath());

ArrayList<String>folderList2=new ArrayList<String>();

folderList2.add(str2+"/"+copyfoldersList[k].getName());

for(int j=0;j<folderList.length;j++){

(new File(folderList2.get(j))).mkdirs(); //如果文件夹不存在 则建立新文件夹

File folders=new File(folderList.get(j));

String[] file=folders.list();

File temp=null;

try {

for (int i = 0; i < file.length; i++) {

if(folderList.get(j).endsWith(File.separator)){

temp=new File(folderList.get(j)+"/"+file[i]);

} else {

temp=new File(folderList.get(j)+"/"+File.separator+file[i]);

}

FileInputStream input = new FileInputStream(temp);

if(temp.isFile()){

FileInputStream input = new FileInputStream(temp);

FileOutputStream output = new FileOutputStream(folderList2.get(j) + "/" + (temp.getName()).toString());

byte[] b = new byte[5120];

int len;

while ( (len = input.read(b)) != -1) {

output.write(b, 0, len);

}

output.flush();

output.close();

input.close();

}

if(temp.isDirectory()){//如果是子文件夹

folderList.add(folderList.get(j)+"/"+file[i]);

folderList2.add(folderList2.get(j)+"/"+file[i]);

}

}

}

catch (Exception e) {

System.out.println("复制整个文件夹内容操作出错");

e.printStackTrace();

}

}

}

}

## 15.移动文件夹

//import java.io.\*;

//import java.util.\*;

LinkedList<String> folderList = new LinkedList<String>();

folderList.add(str1);

LinkedList<String> folderList2 = new LinkedList<String>();

folderList2.add(str2 + str1.substring(str1.lastIndexOf("\\")));

while (folderList.size() > 0) {

(new File(folderList2.peek())).mkdirs(); // 如果文件夹不存在 则建立新文件夹

File folders = new File(folderList.peek());

String[] file = folders.list();

File temp = null;

try {

for (int i = 0; i < file.length; i++) {

if (folderList.peek().endsWith(File.separator)) {

temp = new File(folderList.peek() + File.separator + file[i]);

} else {

temp = new File(folderList.peek() + File.separator + file[i]);

}

if (temp.isFile()) {

FileInputStream input = new FileInputStream(temp);

FileOutputStream output = new FileOutputStream(

folderList2.peek() + File.separator + (temp.getName()).toString());

byte[] b = new byte[5120];

int len;

while ((len = input.read(b)) != -1) {

output.write(b, 0, len);

}

output.flush();

output.close();

input.close();

if (!temp.delete())

System.out.println("删除单个文件操作出错!");

}

if (temp.isDirectory()) {// 如果是子文件夹

for (File f : temp.listFiles()) {

if (f.isDirectory()) {

folderList.add(f.getPath());

folderList2.add(folderList2.peek() + File.separator + f.getName());

}

}

}

}

} catch (Exception e) {

// System.out.println("复制整个文件夹内容操作出错");

e.printStackTrace();

}

folderList.removeFirst();

folderList2.removeFirst();

}

File f = new File(str1);

if (!f.delete()) {

for (File file : f.listFiles()) {

if (file.list().length == 0) {

System.out.println(file.getPath());

file.delete();

}

}

}

## 16.移动一个文件夹下所有的文件夹到另一个目录下

//import java.io.\*;

//import java.util.\*;

File movefolders=new File(str1);

File[] movefoldersList=movefolders.listFiles();

for(int k=0;k<movefoldersList.length;k++){

if(movefoldersList[k].isDirectory()){

ArrayList<String>folderList=new ArrayList<String>();

folderList.add(movefoldersList[k].getPath());

ArrayList<String>folderList2=new ArrayList<String>();

folderList2.add(str2+"/"+movefoldersList[k].getName());

for(int j=0;j<folderList.length;j++){

(new File(folderList2.get(j))).mkdirs(); //如果文件夹不存在 则建立新文件夹

File folders=new File(folderList.get(j));

String[] file=folders.list();

File temp=null;

try {

for (int i = 0; i < file.length; i++) {

if(folderList.get(j).endsWith(File.separator)){

temp=new File(folderList.get(j)+"/"+file[i]);

}

else{

temp=new File(folderList.get(j)+"/"+File.separator+file[i]);

}

FileInputStream input = new FileInputStream(temp);

if(temp.isFile()){

FileInputStream input = new FileInputStream(temp);

FileOutputStream output = new FileOutputStream(folderList2.get(j) + "/" + (temp.getName()).toString());

byte[] b = new byte[5120];

int len;

while ( (len = input.read(b)) != -1) {

output.write(b, 0, len);

}

output.flush();

output.close();

input.close();

temp.delete();

}

if(temp.isDirectory()){//如果是子文件夹

folderList.add(folderList.get(j)+"/"+file[i]);

folderList2.add(folderList2.get(j)+"/"+file[i]);

}

}

}

catch (Exception e) {

System.out.println("复制整个文件夹内容操作出错");

e.printStackTrace();

}

}

movefoldersList[k].delete();

}

}

## 17.以一个文件夹的框架在另一个目录创建文件夹和空文件

//import java.io.\*;

//import java.util.\*;

boolean b=false;//不创建空文件

ArrayList<String>folderList=new ArrayList<String>();

folderList.add(str1);

ArrayList<String>folderList2=new ArrayList<String>();

folderList2.add(str2);

for(int j=0;j<folderList.length;j++){

(new File(folderList2.get(j))).mkdirs(); //如果文件夹不存在 则建立新文件夹

File folders=new File(folderList.get(j));

String[] file=folders.list();

File temp=null;

try {

for (int i = 0; i < file.length; i++) {

if(folderList.get(j).endsWith(File.separator)){

temp=new File(folderList.get(j)+"/"+file[i]);

}

else{

temp=new File(folderList.get(j)+"/"+File.separator+file[i]);

}

FileInputStream input = new FileInputStream(temp);

if(temp.isFile()){

if (b) temp.createNewFile();

}

if(temp.isDirectory()){//如果是子文件夹

folderList.add(folderList.get(j)+"/"+file[i]);

folderList2.add(folderList2.get(j)+"/"+file[i]);

}

}

}

catch (Exception e) {

System.out.println("复制整个文件夹内容操作出错");

e.printStackTrace();

}

}

## 18.复制文件

//import java.io.\*;

int bytesum = 0;

int byteread = 0;

File oldfile = new File(str1);

try {

if (oldfile.exists()) { //文件存在时

FileInputStream inStream = new FileInputStream(oldfile); //读入原文件

FileOutputStream fs = new FileOutputStream(new File(str2,oldfile.getName()));

byte[] buffer = new byte[5120];

int length;

while ( (byteread = inStream.read(buffer)) != -1) {

bytesum += byteread; //字节数 文件大小

System.out.println(bytesum);

fs.write(buffer, 0, byteread);

}

inStream.close();

}

}

catch (Exception e) {

System.out.println("复制单个文件操作出错");

e.printStackTrace();

}

## 19.复制一个文件夹下所有的文件到另一个目录

//import java.io.\*;

File copyfiles=new File(str1);

File[] files=copyfiles.listFiles();

for(int i=0;i<files.length;i++){

if(!files[i].isDirectory()){

int bytesum = 0;

int byteread = 0;

try {

InputStream inStream = new FileInputStream(files[i]); //读入原文件

FileOutputStream fs = new FileOutputStream(new File(str2,files[i].getName());

byte[] buffer = new byte[5120];

int length;

while ( (byteread = inStream.read(buffer)) != -1) {

bytesum += byteread; //字节数 文件大小

System.out.println(bytesum);

fs.write(buffer, 0, byteread);

}

inStream.close();

} catch (Exception e) {

System.out.println("复制单个文件操作出错");

e.printStackTrace();

}

}

}

## 20.提取扩展名

String str2=str1.substring(str1.lastIndexOf(".")+1);