# JAVA檔案

java.io包含IO操作類別：File, InputStream, OutputStream, Reader, Writer和Serializable介面。

## File類別的使用

File

建構子(Java SE, EE)：public File(String pathname);

建構子(Android)：public File(File parent, String child);子路徑可能是檔案或資料夾。

public class IODemo{

public static void main(String[] args){

File file = new File("D:\file.txt");

}

}

新建檔案：public boolean createNewFile() throws IOException;

刪除檔案：public boolean delete();

檔案存在：public boolean exists();

public class IODemo{

public static void main(String[] args){

File file = new File("D:\\file.txt");

if (file.exists()){

file.delete();

}else{

file.createNewFile();

}

}

}

不同OS有不同目錄分割符號，windows用"\"；Linux使用"/"。Java使用分割常數：

public static final String separater;(注意小寫)

public class IODemo{

public static void main(String[] args){

File file = new File("D:"+File.separator+"file.txt");

if (file.exists()){

file.delete();

}else{

file.createNewFile();

}

}

}

若有次目錄，則無法直接新建檔案。

File file = new File("D:" + File.separator + "test" + File.separator + "file.txt");

須先新建次目錄

public class IODemo{

public static void main(String[] args){

File file = new File("D:" + File.separator + "test" + File.separator + "file.txt");

if (!file.getParentFile().exists()){

file.getParentFile().mkdir();

}

file.createNewFile();

}

}

多層次目錄mkdir()無法完成新建，要使用mkdirs()。

檔案長度：public long length();

可讀檔案：public boolean canRead();

可寫檔案：public boolean canWrite();

可執行檔案：public boolean canExecute();

路徑是否為檔案夾：public boolean isDirectory();

路徑是否為檔案：public boolean isFile();

public class IODemo{

public static void main(String[] args){

File file = new File("D:" + File.separator + "file.txt");

if (file.exists()){

System.out.println("檔案長度：" + file.length());

System.out.println("檔案可讀：" + file.canRead());

System.out.println("檔案可寫：" + file.canWrite());

System.out.println("檔案可執：" + file.canExecute());

System.out.println("是否檔案：" + file.isFile());

System.out.println("是否檔案夾：" + file.isDirectory());

System.out.println("是否檔案夾：" + file.getParentFile().isDirectory());

}

}

}

印出目錄中的內容：public String[] list();

印出目錄中的檔案：public String[] listFiles();

public class IODemo{

public static void main(String[] args){

File file = **new** File("D:" + File.***separator***);

**if** (file.isDirectory()){

String[] str1 = file.list();

**for** (**int** x=0; x<str1.length; x++){

System.***out***.println(str1[x]);

}

File[] str2 = file.listFiles();

**for** (**int** x=0; x<str2.length; x++){

System.***out***.println(str2[x]);

}

}

}

}

要印出所有檔案，必須使用遞迴程式。

public class IODemo{

public static void main(String[] args){

File file = new File("D:" + File.separator);

list(file);

}

public static void list(File file){

if (file.isDirectory()){

File[] f = file.listFiles();

if (f!=null){//防止空目錄產生錯誤

for (int x=0; x<f.length; x++){

list(f[x]);

}

}

}else{

System.out.println(file);

}

}

}

File類別不支授檔案內容的操作。

## 位元組操作

### OutputStream

public abstract class OutputStream

extends Object

implements Closeable, Flushable

可以關閉。

public interface Closeable{

public void close() throws IOException;

}

可以刷新。

public interface Flushable{

public void flush() throws IOException;

}

輸出全部位元組資料：public void write(byte[] b) throws throws IOException;

輸出部分位元組資料：public void write(byte[] b, int off, int len) throws throws IOException;

輸出單一位元組資料：public abstract void write(int b) throws throws IOException;

OutputStream是抽象類別，要用子類別產生物件。要操作檔案使用子類別FileOutputStream。

public FileOutputStream(File file) throws FileNotFoundException;

public class IODemo{

public static void main(String[] args){

File file = new File("D:" + File.separator+"test"+ File.separator+"file.txt");

if (!file.getParentFile().exists()){

file.getParentFile().mkdir();

}

OutputStream out = new FileOutputStream(file);//轉型

String str = "Hello World";

byte[] data = str.getBytes();

out.write(data);

out.close();

}

}

新檔案換舊檔案；檔案不存在會新建。

### InputStream

讀取資料到陣列，回傳讀取個數：public int read(byte[] b) throws IOException;

讀取部分資料到陣列，回傳讀取個數：public int read(byte[] b, int off, int len) throws IOException;

InputStream是抽象類別，要實體化物件必須使用子類別FileInputStream。

建構子：public FileInputStream(File file) throws FileNotFoundException;

public class IODemo{

public static void main(String[] args){

File file = new File("D:" + File.separator+"test"+ File.separator+"file.txt");

if (file.exists()){

InputStream in = new InputStream(file);

byte[] data = new byte[1024];

int len = in.read(data);

in.close();

System.out.println("讀取資料：["+new String(data, 0, len) +"]");

}

}

讀取單個位元組：public abstract int read() throws IOException;

每一次讀取一個位元組，並回傳下一個位元組位址；讀到檔案結尾，回傳-1。

public class IODemo{

public static void main(String[] args){

File file = new File("D:" + File.separator+"test"+ File.separator+"file.txt");

if (file.exists()){

InputStream in = new InputStream(file);

StringBuffer buf = new StringBuffer();

int temp = 0;

do{

temp = in.read()

if (temp != -1) buf.append((char) temp);

}while (temp != -1);

in.close();

System.out.println("讀取資料：["+buf+"]");

}

}

}

public class IODemo{

public static void main(String[] args){

File file = new File("D:" + File.separator+"test"+ File.separator+"file.txt");

if (file.exists()){

InputStream in = new InputStream(file);

StringBuffer buf = new StringBuffer();

int temp = 0;

while (temp = in.read()) != -1) buf.append((char) temp);

in.close();

System.out.println("讀取資料：["+buf+"]");

}

}

}

## 字元操作

### Writer

public abstract class Writer

extends Object

implements Appendable, Closeable, Flushable

輸出字串：public void write(String str) throws IOException;

輸出字元陣列：public void write(char[] cbuf) throws IOException;

public class IODemo{

public static void main(String[] args){

File file = new File("D:" + File.separator+"test"+ File.separator+"file.txt");

if (!file.getParentFile().exists()){

file.getParentFile().mkdir();

}

Writer out = new FileWriter(file);//轉型

String str = "Hello World";

out.write(str);

out.close();

}

}

### Reader

public abstract class Reader

extends Object

implements Closeable, Flushable

讀取單個字元：public int read() throws IOException;

讀取內容到字元陣列：public int read(char[] cbuf) throws IOException;

讀取部分內容到字元陣列：public abstract int read(char[] cbuf, int off, int len) throws IOException;

Reader是抽象類別，要實體化物件要用FileReader。

public class IODemo{

public static void main(String[] args){

File file = new File("D:" + File.separator+"test"+ File.separator+"file.txt");

if (file.exists()){

Reader in = new FileReader(file);

char[] data = new char[1024];

int len = in.read(data);

in.close();

System.out.println("讀取資料：["+new String(data, 0, len)+"]");

}

}

}

## 位元組(byte)和字元(char)

文本檔案使用字元為單位；其他(網路、圖像、音訊…)使用字元為單位。字元在處理中文比位元組方便。

OutputStream即使沒有關閉檔案，也OK。一般資料的傳輸以此為主。

Writer在結束時一定要關閉檔案，因為Writer將資料儲存在緩衝區。須用close()或flush()將資料清空。

public class copy{

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

if (args.length !=2){

System.out.println("指令錯誤");

System.exit(1);

}

File inFile = new File(args[0]);

if (!inFile.exists()){

System.out.println("檔案不存在");

System.exit(1);

}

File outFile = new File(args[1]);

if (!outFile.getParentFile().exists()){

outFile.getParentFile().mkdirs();

}

InputStream input = new FileInputStream(inFile);

OutputStream output = new FileOutputStream(outFile);

int temp =0;

while ((temp = input.read()) != -1) output.write(temp);

input.close();

output.close();

}

}

每次操作多位元組。

public class copy{

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

if (args.length !=2){

System.out.println("指令錯誤");

System.exit(1);

}

File inFile = new File(args[0]);

if (!inFile.exists()){

System.out.println("檔案不存在");

System.exit(1);

}

File outFile = new File(args[1]);

if (!outFile.getParentFile().exists()){

outFile.getParentFile().mkdirs();

}

byte[] data = new byte[1024];

InputStream input = new FileInputStream(inFile);

OutputStream output = new FileOutputStream(outFile);

int temp =0;

while ((temp = input.read(data)) != -1) output.write(data,0 ,temp);

input.close();

output.close();

}

}

## Byte to Char

輸出位元組(Byte)流轉成字元流：OutputStreamWriter

輸入位元組(Byte)流轉成字元流：InputStreamReader

OutputStreamWriter建構子：public OutputStreamWriter(OutputStream out);

InputStreamReader建構子：public InputStreamReader(InputStream in);

public class IODemo{

public static void main(String[] args){

File file = new File("D:" + File.separator+"test"+ File.separator+"file.txt");

if (!file.getParentFile().exists()){

file.getParentFile().mkdir();

}

OutputStream out = new FileOutputStream(file);

Writer w = new OutputStreamWriter(out);

String str = "Hello World";

w.write(str);

w.close();

out.close();//關閉有順序

}

}