Creazione, eliminazione, lettura e scrittura di file di testo

Java mette a disposizione degli sviluppatori diverse classi per lavorare con i file di testo, analizziamo le principali:

- `java.io.File` – La classe File è una rappresentazione astratta di un file o di una directory. Permette di lavorare a livello più alto, come ad esempio creare un file vuoto, eliminarlo, ricercarlo o rinominarlo.
- `java.io.FileReader` – La classe FileReader permette di leggere, uno alla volta, i caratteri contenuti in un file di testo.
- `java.io.BufferedReader` – La classe BufferedReader è simile alla classe FileReader ma permette di leggere i caratteri presenti nel file in blocchi. I caratteri letti vengono memorizzati in un buffer temporaneo. Quando i dati vengono richiesti, vengono letti dal buffer, quindi le prestazioni migliorano notevolmente non dovendo ogni volta avere l'ok del sistema operativo per accedere al file system.
- `java.io.FileWriter` – La classe FileWriter permette di scrivere i caratteri in un file di testo uno alla volta.
- `java.io.BufferedWriter` – La classe BufferedWriter è simile alla classe FileWriter ma permette di scrivere i caratteri nel file in blocchi. I caratteri vengono memorizzati temporaneamente in un buffer temporaneo. Periodicamente i caratteri vengono letti dal buffer e scritti fisicamente sul file, quindi, anche in questo caso, le prestazioni migliorano notevolmente.
- `Java.io.PrintWriter` – La classe PrintWriter è simile alla classe BufferedWriter ma permette di scrivere nel file stringhe formattate.

Per prendere confidenza con queste classi, analizziamo dei semplicissimi esempi:

Listato 1. Creazione di un file vuoto

```
public static void newFile() {
  String path = "C:/html.txt"; /* si assegna un percorso al file,
                                * diversamente il file creato viene creato nella stessa cartella in cui
                                * vengono salvati gli altri file di eclipse,
                                * si consiglia di non indicare il percorso finché non si avrà buona
                                 * padronanza nella gestione dei file ,  tenendo ben presente che può
                                 * sempre essere fatto*/
  try {
    File file = new File(path); // potrebbe generare errore nel caso non si indichi il percorso completo

    if (file.exists())
      System.out.println("Il file " + path + " esiste");
    else if (file.createNewFile())
      System.out.println("Il file " + path + " è stato creato");
    else
      System.out.println("Il file " + path + " non può essere creato");
  }
  catch (IOException e) {
    e.printStackTrace();
  }
}
```

Il costruttore della classe File riceve in ingresso il path del file. La creazione di un'istanza di File non genera la creazione fisica del file sul disco fisso; occorre chiamare il metodo `createNewFile()` per crearlo fisicamente.

Se proviamo a lanciare l'esempio, la prima volta il file verrà creato, la seconda volta invece verrà stampato che il file è già presente poiché il metodo exists restituirà true.

I metodi principali della classe File sono i seguenti:

- `exists()` – che restituisce true se il file esiste, false altrimenti.
- `createNewFile()` – che crea effettivamente il file sul disco.
- `delete()` – che elimina il file dal disco.
- `isFile()` – che restituisce true se si tratta di un file, false altrimenti.
- `isDir()` – che restituisce true se si tratta di una directory, false altrimenti.
- `mkDir()` – che permette di creare una directory.

Alcune operazioni possono generare diverse eccezioni di I/O qualora per qualche motivo il file non può essere creato, eliminato, rinominato o altro.

Listato 2. Scrittura di un file - “simile all’esempio da noi proposto”

```
public static void writeFile() {
  String path = “C:/html.txt”;
  try {
    File file = new File(path);
    FileWriter fw = new FileWriter(file);
    fw.write(“Questo è il nostro primo file”);
    fw.flush();
    fw.close();
  }
  catch(IOException e) {
    e.printStackTrace();
  }
}
```

il costruttore della classe FileWriter riceve in ingresso un’istanza della classe File.

I metodi principali della classe FileWriter sono i seguenti:

- write(String) – scrive la stringa sul file.
- close() – chiude il descrittore del file.

Come abbiamo detto in precedenza la classe FileWriter scrive un singolo carattere alla volta. Per ottimizzare l’operazione, possiamo utilizzare la classe BufferedWriter. Per il programmatore non cambia molto ma sicuramente viene ottenuto un vantaggio in termini di prestazioni. Il costruttore della classe BufferedWriter riceve in ingresso un’istanza della classe FileWriter. Il metodo per scrivere è sempre `write()`.

Listato 3. Scrittura di un file con BufferWriter

```
public static void writeFile2() {
  String path = “C:/html.txt”;
  try {
    File file = new File(path);
    FileWriter fw = new FileWriter(file);
    BufferedWriter bw = new BufferedWriter(fw);
    bw.write(“Questo è il nostro primo file”);
    bw.flush();
    bw.close();
  }
  catch(IOException e) {
    e.printStackTrace();
  }
}
```

Listato 4. Lettura di un file di testo

```
public static void readFile() {
  String path = "C:/html.txt";
  char[] in = new char[50];
  int size = 0;
  try {
    File file = new File(path);
    FileReader fr = new FileReader(file);
    size = fr.read(in);

    System.out.print("Caratteri presenti: " + size + "\n");

    System.out.print("Il contenuto del file è il seguente:\n");

    for(int i=0; i<size; i++)
      System.out.print(in[i]);

    fr.close();
  }
  catch(IOException e) {
    e.printStackTrace();
  }
}
```

Il costruttore della classe FileReader riceve in ingresso un'istanza della classe File.

I metodi principali per della classe sono i seguenti:

- read(char[n]) – legge n caratteri dal file, uno alla volta, a partire dalla posizione corrente e li memorizza nell'array di caratteri passato in ingresso. Il metodo restituisce, inoltre, il numero di caratteri letti.
- close() – chiude il descrittore del file.

Così come abbiamo fatto per la scrittura, anche per la lettura possiamo ottenere vantaggi in termini di prestazioni, utilizzando la classe BufferedReader. Il costruttore della classe BufferedReader riceve in ingresso un'istanza della classe FileReader. Il metodo da utilizzare per leggere è sempre `read()`.

Listato 5. Lettura di un file di testo con BufferRead

```
public static void readFile2() {
  String path = “C:/html.txt”;
  char[] in = new char[50];
  int size = 0;
  try {
    File file = new File(path);
    FileReader fr = new FileReader(file);
    BufferedReader br = new BufferedReader(fr);
    size = br.read(in);

    System.out.print(“Caratteri presenti: ” + size + “\n”);

    System.out.print(“Il contenuto del file è il seguente:\n”);

    for(int i=0; i<size; i++)
      System.out.print(in[i]);

    br.close();
  }
  catch(IOException e) {
    e.printStackTrace();
  }
}
```

Listato 6. Altro esempio per la lettura del contenuto in un file

```
import java.io.*;
import java.net.*;

// definiamo la classe principale
public class InfoFile
{
  public static void main(String[] args)
  {
    // definiamo il percorso al file da leggere
    File doc=new File("C:/doc.txt");
    URL path=null;

    // creaiamo un blocco try-catch per intercettare le eccezioni
    try
    {
      // mostriamo il percorso al file
      path=doc.toURL();
      System.out.println("Il doc si trova nel percorso" + path);

      / /mostriamo il nome del file
      doc=new File(path.getFile());
      System.out.println("Nome del file " + doc);
      int i;

      // apriamo lo stream di input...
      InputStream is=path.openStream();
      BufferedReader br=new BufferedReader(new InputStreamReader(is));

      // ...e avviamo la lettura del file con un ciclo
      do
      {
        i=br.read();
        System.out.println((char)i);
      }
      while (i!=-1);
      is.close();
    }

    // intercettiamo eventuali eccezioni
    catch (MalformedURLException e)
    {
      System.out.println("Attenzione:" + e);
    }
    catch (IOException e)
    {
      System.out.println(e.getMessage());
    }
  }
}
```

N.B.
grazie alla classe File() un file partecipa al flusso di dati in un'applicazione esattamente come qualsiasi altra informazione o dato.

All’indirizzo: http://docs.oracle.com/javase/7/docs/api/java/io/File.html
Si può trovare la documentazione ufficiale (in lingua inglese) dell’Oracle sulla classe File del linguaggio Java.

All’indirizzo: http://tutorials.jenkov.com/java-io/file.html
È presente un tutorial sulla gestione dell’input/output da file.