**Seleccionar archivos con JFileChooser**

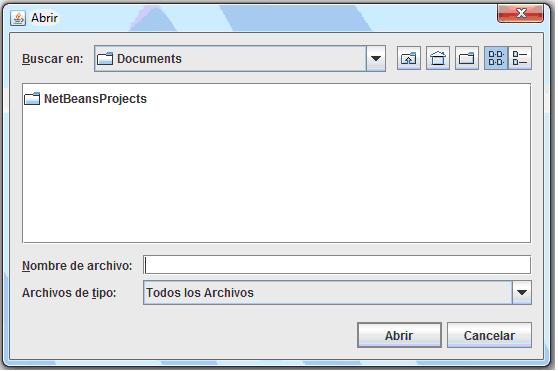
**SELECCIONAR ARCHIVOS DE FORMA GRÁFICA UTILIZANDO LA CLASE JAVA JFILECHOOSER**

Cuando estamos trabajando con aplicaciones java que tienen que acceder a archivos, en lugar de indicar la ruta podemos realizar la selección del archivo mediante la clase **JFileChooser**.

Esta clase nos permite acceder de forma gráfica al sistema de archivos del ordenador y navegar por él para seleccionar el archivo que necesita nuestra aplicación.

La clase JFileChooser se encuentra en el paquete javax.swing.

Al utilizar JFileChooser se mostrará una ventana similar a esta:

[](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEibzvYMo7fproVYF4JhiiJ2zCToXAUkJO4Q2nLsa16Xp8S_oMuI0XakRsLciBuYfQuooa25tsDMtX-17Tj1MVapvQTyLBcQGGbHUjA1feOxSwLkyf5jreRoSBP01wev9psiWn2VoHfyVnWt/s1600/JFileChooser.jpg)

Navegando a través de ella podremos seleccionar el archivo que necesite procesar nuestra aplicación.

**Ejemplo** de uso de la clase JFileChooser:

En este ejemplo usamos JFileChooser para seleccionar un archivo de texto. A continuación, se lee el archivo seleccionado línea a línea y se muestra por pantalla su contenido. Para realizar la lectura del fichero se utiliza la clase Scanner.

import java.io.File;

import java.io.FileNotFoundException;

import java.util.Scanner;

import javax.swing.JFileChooser;

public class Main {

public static void main(String[] args) {

Scanner entrada = null;

JFileChooser fileChooser = new JFileChooser();

fileChooser.showOpenDialog(fileChooser);

try {

String ruta = fileChooser.getSelectedFile().getAbsolutePath();

File f = new File(ruta);

entrada = new Scanner(f);

while (entrada.hasNext()) {

System.out.println(entrada.nextLine());

}

} catch (FileNotFoundException e) {

System.out.println(e.getMessage());

} catch (NullPointerException e) {

System.out.println("No se ha seleccionado ningún fichero");

} catch (Exception e) {

System.out.println(e.getMessage());

} finally {

if (entrada != null) {

entrada.close();

}

}

}

}

Explicamos el código:

Primero se crea el objeto JFileChooser invocando al constructor de la clase:

JFileChooser fileChooser = new JFileChooser();

A continuación el método showOpenDialog() muestra la ventana donde podremos seleccionar el fichero a leer:

fileChooser.showOpenDialog(fileChooser);

La ventana inicial muestra por defecto la carpeta *Documents*. Para cambiar esta carpeta y que la ventana de selección se abra en una carpeta diferente debemos indicar la ruta en el constructor:

JFileChooser fileChooser = new JFileChooser(ruta);

Por ejemplo, para mostrar el contenido de una carpeta llamada *ficheros* que se encuentra en la unidad C: escribiríamos:

JFileChooser fileChooser = new JFileChooser("C:/ficheros");

También se puede utilizar el método setCurrentDirectory. Por ejemplo para indicar que la carpeta a mostrar es c:/ficheros lo indicaremos así:

File nuevaRuta = new File("C:/ficheros ");

fileChooser.setCurrentDirectory(nuevaRuta);

A continuación, mediante la instrucción:

String ruta = fileChooser.getSelectedFile().getAbsolutePath()

se obtiene un String que contiene la ruta del fichero seleccionado.

Una vez obtenida la ruta del archivo se crea un Scanner que hemos llamado *entrada* para realizar la lectura:

File f = new File(ruta);

entrada = new Scanner(f);

Finalmente el bucle *while* se encarga de leer el fichero de texto línea a línea y mostrar cada línea del mismo por pantalla.

Respecto a las excepciones capturadas:

* FileNotFoundException: La lanza la clase Scanner si no se encuentra el fichero en la ruta indicada.
* NullPointerException: En el caso de no haber seleccionado ningún fichero, porque se ha pulsado el botón cancelar o se ha cerrado la ventana directamente, se lanzará la excepción NullPointerException que es necesario capturar para que el programa termine de forma correcta.

En lugar de capturar la excepción NullPointerException, podemos utilizar el valor devuelto por el método showOpenDialog para saber si se ha seleccionado o no algún archivo.

El método showOpenDialog devuelve los valores:

·         JFileChooser.CANCEL\_OPTION si se ha cancelado la acción.

·         JFileChooser.APPROVE\_OPTION si se ha seleccionado un archivo.

·         JFileChooser.ERROR\_OPTION si ha ocurrido un error o se ha cerrado la ventana de diálogo.

En este caso el ejemplo propuesto quedaría así:

import java.io.File;

import java.io.FileNotFoundException;

import java.util.Scanner;

import javax.swing.JFileChooser;

public class Main {

public static void main(String[] args) {

Scanner entrada = null;

JFileChooser fileChooser = new JFileChooser();

int valor = fileChooser.showOpenDialog(fileChooser);

if (valor == JFileChooser.APPROVE\_OPTION) {

String ruta = fileChooser.getSelectedFile().getAbsolutePath();

try {

File f = new File(ruta);

entrada = new Scanner(f);

while (entrada.hasNext()) {

System.out.println(entrada.nextLine());

}

} catch (FileNotFoundException e) {

System.out.println(e.getMessage());

} finally {

if (entrada != null) {

entrada.close();

}

}

} else {

System.out.println("No se ha seleccionado ningún fichero");

}

}

}

Si se ejecuta el programa se puede comprobar que el usuario solo puede seleccionar archivos. Mediante JFileChooser podemos limitar al usuario para que solo pueda seleccionar directorios o archivos o ambas cosas utilizando el método setFileSelectionMode(modo)

Como parámetro en modo podemos usar estos valores:

JFileChooser.DIRECTORIES\_ONLY : permite seleccionar solo directorios

JFileChooser.FILES\_ONLY : permite seleccionar solo archivos. Es el valor por defecto.

JFileChooser.FILES\_AND\_DIRECTORIES : Permite seleccionar archivos y directorios.

**USO DE FILTROS**

En algunas ocasiones resulta útil que en la ventana no se muestren todos los archivos sino que mediante un filtro solo se muestren los archivos que nos sean útiles para nuestra aplicación, por ejemplo los que tengan una determinada extensión (.java, .txt, etc.).

Para ello se utiliza la clase **FileNameExtensionFilter**.

Ejemplo de uso de FileNameExtensionFilter: Vamos a utilizar la clase FileNameExtensionFilter para indicar a JFileChooser que solo muestre los archivos con extensión .java

import java.io.File;

import java.io.FileNotFoundException;

import java.util.Scanner;

import javax.swing.JFileChooser;

import javax.swing.filechooser.FileFilter;

import javax.swing.filechooser.FileNameExtensionFilter;

public class Main {

public static void main(String[] args) {

Scanner entrada = null;

//Se crea el JFileChooser. Se le indica que la ventana se abra en el directorio actual

JFileChooser fileChooser = new JFileChooser(".");

//Se crea el filtro. El primer parámetro es el mensaje que se muestra,

//el segundo es la extensión de los ficheros que se van a mostrar

FileFilter filtro = new FileNameExtensionFilter("Archivos java (.java)", "java");

//Se le asigna al JFileChooser el filtro

fileChooser.setFileFilter(filtro);

//se muestra la ventana

int valor = fileChooser.showOpenDialog(fileChooser);

if (valor == JFileChooser.APPROVE\_OPTION) {

String ruta = fileChooser.getSelectedFile().getAbsolutePath();

try {

File f = new File(ruta);

entrada = new Scanner(f);

while (entrada.hasNext()) {

System.out.println(entrada.nextLine());

}

} catch (FileNotFoundException e) {

System.out.println(e.getMessage());

} finally {

if (entrada != null) {

entrada.close();

}

}

} else {

System.out.println("No se ha seleccionado ningún fichero");

}

}

}