Práctica VIII - File Chooser (JFileChooser)

Siempre que queramos acceder a un archivo de forma gráfica podemos usar `JFileChooser`, el cuál mostrará al usuario una ventana para buscar el archivo. Podemos utilizar `JFileChooser` para abrir un archivo, para abrir un directorio, para guardar un archivo, entre otros usos.

Crear un `JFileChooser`

Lo más recomendable es declarar la instancia de `JFileChooser` como un atributo de la clase `JFrame` para poder utilizarlo dentro de toda la clase. El código debería ser similar al siguiente:

```
public class MiVentana extends javax.swing.JFrame {

    JFileChooser fileChooser = new JFileChooser();

    public MiVentana() {
        initComponents();
    }

    ...
}
```

> **Nota:** puedes modificar a `JFileChooser` dentro del contructor de la clase.

Mostrar un diálogo para abrir un archivo

Dentro de algún evento (como al pulsar un botón) podemos abrir una ventana de diálogo para abrir un archivo usando `fileChooser.showOpenDialog(this)`. El método `showOpenDialog` requiere de un componente padre para iniciar la ventana, generalmente se usa `this` lo cual significa que nuestra ventana derivada de `JFrame` será el padre de la ventana de diálogo, pero igual se podría poner un botón u otro componente.

El método `showOpenDialog` devuelve un entero (`int`) con el estatus de apertura que puede ser `JFileChooser.APPROVE_OPTION`, `JFileChooser.CANCEL_OPTION` o `JFileChooser.ERROR_OPTION`, por lo que mediante una estructura condicional podemos determinar si el resultado devuelto fue alguna de esas opciones. El código para controlar la opción debería ser similar a:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    int result = this.fileChooser.showOpenDialog(this);

    switch(result) {
        case JFileChooser.APPROVE_OPTION:
            // Archivo seleccionado
            break;
        case JFileChooser.CANCEL_OPTION:
            // Selección cancelada
            break;
        case JFileChooser.ERROR_OPTION:
            // Error durante la selección del archivo
            break;
    }
}
```

> **Nota:** Observa que se utilizó una estructura condicional `switch` pero deberías ser capaz de utilizar la estructura `if-else` sin mayor problema.

El fragmento de código anterior controla el evento de un botón, el cual abre la ventana para abrir un archivo. Luego el resultado de la venta es recuperado y guardado en la variable `int result`. Finalmente mediante `switch` se compara si `result == JFileChooser.APPROVE_OPTION` o los otros. En el caso se que sea igual por ejemplo a `JFileChooser.APPROVE_OPTION` significaría que en la ventana se seleccionó un archivo y se pulsó el botón `Abrir` de la ventana de diálogo.

Recuperar el archivo seleccionado en `JFileChooser`

En el caso que se haya generado `JFileChooser.APPROVE_OPTION`, podemos recuperar el archivo seleccionado, el cuál será parte de la clase `File`, es decir, `NO SE ABRE EL ARCHIVO HASTA QUE SE DEFINA UN FileInputStream`. Para recuperar el archivo (descriptivo) usamos `fileChooser.getSelectedFile()` o `fileChooser.getSelectedFiles()` dependiendo si se requiere el archivo seleccionado o los archivos seleccionados. El código sería similar a:

```
switch(result) {
    case JFileChooser.APPROVE_OPTION:
        // Archivo seleccionado
        File f = this.fileChooser.getSelectedFile();
        System.out.println(f.getAbsolutePath());
        break;
    case JFileChooser.CANCEL_OPTION:
        // Selección cancelada
        break;
    case JFileChooser.ERROR_OPTION:
        // Error durante la selección del archivo
        break;
}
```

Observa que `File f` sólo recupera la información del archivo, como su tamaño y su ruta. Si requerimos abrir el contenido del archivo como tal, deberíamos hacer `FileInputStream in = new FileInputStream(f);` .

Elegir el modo de selección de archivos y directorios

Podemos definir que tipo de selección puede hacer el usuario mediante `fileChooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY)` , otras opciones son `JFileChooser.FILES_ONLY` y `JFileChooser.FILES_DIRECTORIES` .

Definir los filtros de archivos seleccionables

Podemos definir que tipos de archivos leer el `JFileChooser` mediante `fileChooser.addChoosableFileFilter(filter)` donde `filter` es un objeto de la clase `FileNameExtensionFilter` , por ejemplo, para ver archivos de imagen tendríamos y otro filtro para ver documentos:

```
FileNameExtensionFilter filter = new FileNameExtensionFilter("Imágenes", "jpg", "png", "tif", "

fileChooser.addChoosableFileFilter(filter);

FileNameExtensionFilter filter2 = new FileNameExtensionFilter("Documentos", "txt", "doc", "doc

fileChooser.addChoosableFileFilter(filter2);
```

Problemas

- Crea un `JFileChooser` llamado `fc` como atributo de la clase `JFrame`.
- Crea un `JButton` llamado `abrir` y un `JList` con un `DefaultListModel<String>` vacío.
- En el evento del botón `abrir` recupera el archivo de tipo `File` mediante `fc.getSelectedFile()` sólo si el resultado de la ventana de diálogo fue `JFileChooser.APPROVE_OPTION`.
- Almacena la ruta absoluta del archivo mediante `model.addElement(f.getAbsolutePath())`.

Diplomado de Java - Alan Badillo Salas (badillo.soft@hotmail.com)
Instituto Politécnico Nacional - Centro de Investigación en Cómputo