¿Qué es la biblioteca Swing?

Luis.puig
1 de Agosto

Con la evolución de la tecnología y el surgimiento del concepto de multiplataforma con la versión 1.2 de Java (anteriormente llamado Playground) o Java 2, en el año 1998 surge la API gráfica de Swing **Java Swing** que integra la JFC (Java Foundation Classes) las cuáles reúnen componentes para la construcción de una GUI (Graphical User Interface - Interface Gráfica de Usuario).

La biblioteca posibilita el desarrollo de interfaces elaboradas para un ambiente de computación heterogéneo con interacción más agradable. Las aplicaciones pueden seguir una apariencia y comportamiento de la plataforma nativa de Java o alguna plataforma personalizada. Permite el gerenciamiento de layouts, manejo de eventos, manipulación de imágenes en dos dimensiones - 2D y abarca diversos idiomas, entre otras.

Además de esto, extiende el beneficio de computación con Java y la API de accesibilidad que funciona con dispositivos de entradas y salidas, como lectores de pantalla, terminales en Braille y otros. Sus componentes son independientes de la plataforma con el patrón **MCV (Modelo-Vista-Controlador).**

Antes de presentar qué es un componente en el contexto de **Java Swing**, vale la pena mencionar que **Swing** emplea en su estructura conceptos para prácticas de programación con el paradigma orientado a objetos: **Clases, Objetos, Encapsulación, Herencia y Polimorfismo, etc.**

Componentes

En la programación orientada a objetos los componentes son **Clases/Objetos** que implementan la interfaz gráfica de usuario. Los componentes básicos proporcionados por la biblioteca **Java Swing** son:

- JButton
- JCheckBoxName
- JLabelGenericName
- JTextAreaName
- JTextFieldName
- JPasswordField
- JRadioButtonName
- JOptionPane

Etiqueta (JLabel)

Estos componentes puede ser usado para mostrar textos simples, imágenes con textos. También pueden ser usados para mostrar los resultados de un proceso. Los métodos más usados son:

- `setText()` - Permite colocar un valor de texto, puede ser usado para mostrar un resultado en este componente.
- `setFont()` - Permite definir el tipo y el estilo de la fuente (negrita y/o cursiva) y el tamaño de la fuente.
- `setIcon()` - Permite que un ícono o una imagen sea usada y mostrada en la pantalla.

Cajas de Texto (JTextField, JPasswordField y JTextArea)

Estos componentes son usados para que el usuario introduzca un valor textural y capturarlo. `JTextField` permite introducir una línea de texto, `JPasswordField` permite introducir un valor pero lo muestra camuflado para no poder ver el valor introducido y `JTextArea` permite introducir varias líneas de texto. Los métodos más usados son:

- `setText()` - Permite colocar un valor de texto, puede ser usado para mostrar un resultado en este componente.
- `getText()` - Permite recuperar el valor de una caja de texto.
- `setFont()` - Permite definir el tipo y el estilo de la fuente (negrita y/o cursiva) y el tamaño de la fuente.
- `setEnabled()` - Permite que la escritura sea deshabilitada y puede ser usado para mostrar un resultado co en un JLabel.

Botones (JRadioButton, JCheckBox y JButton)

Estos componentes permiten varias formas gráficas con las cuales el usuario puede interactuar, el `JRadioButton` permite elegir una opción dentro de un grupo de elecciones, en cuanto al `JCheckBox` permite seleccionar más de una de estas opciones. Finalmente los `JButton` son usados para ejecutar un evento cuando son presionados.

Listas Desplegables (JComboBox)

Parecido al `JRadioButton`, permite seleccionar una opción de un grupo de alternativas pero sin ocupar espacio en la pantalla.

Cajas de Diálogo (JOptionPane)

Este permite mostrar una caja con un mensaje, un icono y botones. Un código de ejemplo sería:

```
JOptionPane.showMessageDialog(null, “Mensaje”, “Título”, JOptionPane.WARNING_MESSAGE);
```

El segundo y tercer parámetro indican el mensaje y el título, estas cajas pueden ser usadas para mostrar mensajes de confirmación, errores, avisos entre otras posibilidades.

Jerarquía de Componentes

Ahora que ya vimos sobre los componentes, podemos hablar sobre la jerarquía existente entre ellos dentro de la biblioteca **Swing**. Este rango es utilizado en la funcionalidad de estos elementos para la comprensión de una aplicación **Swing** y es comúnmente visualizada como un árbol de componentes.

Este árbol trabaja con tres tipos de elementos en una aplicación:

1. **Contenedor de nivel superior**: Es un elemento que generalmente se usa como base, es decir, proporcionar lugar para usar otros elementos. Ejemplos de este tipo serían: `JFrame, JDialog y JApplet`.
2. **Componente de nivel intermedio**: Es un elemento que se usa solo para manejar la ubicación de los elemen de botones y etiquetas. Ejemplos de este tipo serían: `JPanel, JScrollPane y JTabbedPane.`
3. **Contenedor Atómico**: Es un elemento que no se usa para almacenar otros elementos, es una entidad autosuficiente, que sirve para presentar información al usuario o para recibir información proporcionada por usuario. Algunos de los ejemplos de este tipo son: `JButton, JLabel, JComboBox, JTextField y JTal`

Al utilizar JFrame, este componente acaba creando otros tres componentes, estos de tipo intermedio, se utilizan para configuraciones avanzadas de la interfaz gráfica: **Root Pane, Layered Pane y Glass Pane**. Este artículo fue diseñado para presentar **Swing**, por lo que no profundizaremos en estos componentes.

Considerando este diagrama, podemos crear una ventana con **Swing** como lo demuestra el siguiente ejemplo práctico:

```
import java.awt.GridBagLayout;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextArea;

public class Main {
  public static void main(String[] args){

    // componente JFrame

   JFrame miJFrame = new JFrame("Ejemplo - Java Swing");
    miJFrame.setSize(500,300);

    // componente JPanel`

    `JPanel miJPanel = new JPanel();`

    `miJPanel.setSize(300, 300);`

    // usamos este diseño para centrar los componentes de JPanel`

    miJPanel.setLayout(new GridBagLayout());`

    // componente JTextField`

    JLabel miJLabel = new JLabel();

    miJLabel.setText("Dime tu opinión acerca de Java Swing:  ");

    // componente JTextArea

    JTextArea miJTextArea = new JTextArea(5,20);

    // conecta los componentes JLabel y JTextField en JPanel`

    miJPanel.add(miJLabel);

    miJPanel.add(miJTextArea);
```

```
    // conectar Jpanel a JFrame`

    miJFrame.add(miJPanel);

    // hacer visible JFrame

    miJFrame.setVisible(true);

  }

}
```

Resultado del código:

Conclusión

Este es un ejemplo de una interfaz gráfica con **Java Swing**, creamos los componentes de forma individual, uno a la vez, definiendo sus características principales y al final conectamos todos los componentes para poder ver el resultado en nuestra pantalla. En general una aplicación **GUI - Swing** respeta las siguientes etapas:

- Creación de la ventana que contendrá los demás objetos gráficos de la aplicación
- Inserción de los componentes de la interfaz
- Manejo de eventos
- Lógica del programa

Ahora que ya sabes la existencia de los componentes **Swing** es super recomendable buscar y explorar más información sobre esta biblioteca. Son innumerables las posibilidades con **Java Swing**. Sus componentes son divertidos para jugar y pueden ayudarte a construir aplicaciones geniales. Por lo tanto, prueba estos componentes incluyéndolos en tu aplicación **Swing**.