ACTIVIDAD GUIADA 15

Usando la interfaz gráfica de Netbeans

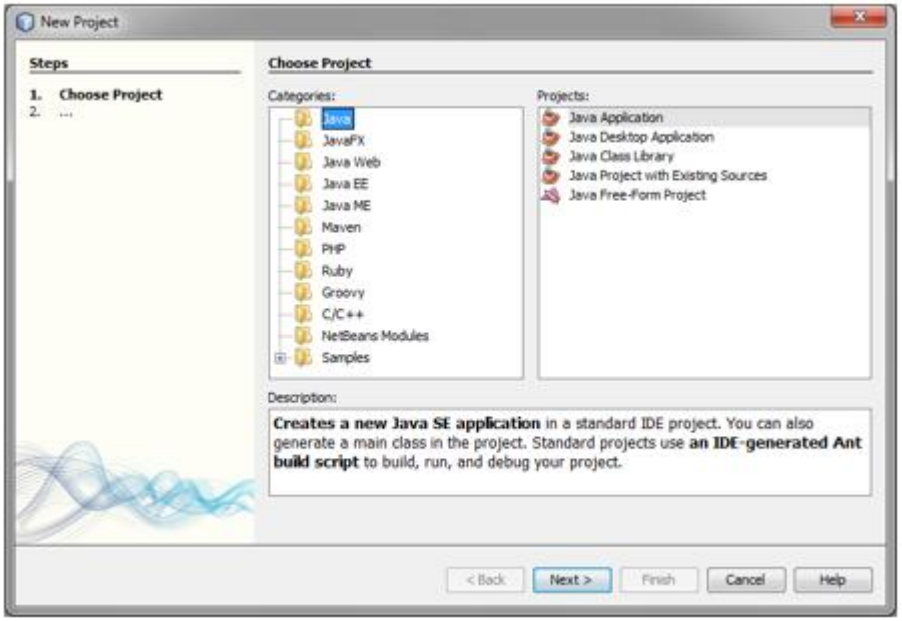
El constructor de interfaces de usuario de NetBeans (*NetBeans GUI Guilder*, antes llamado *Matisse*) permite crear interfaces de usuario de manera simple mediante el uso de componentes *Drag & Drop*, es decir, arrastrando y soltando.

El siguiente documento ilustra acerca del uso básico del mismo.

Creación de un proyecto

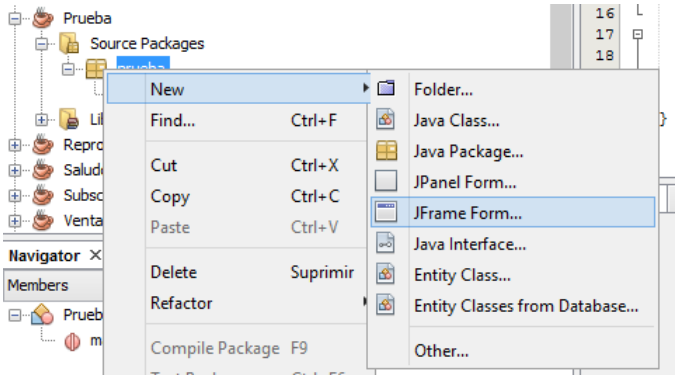
El primer paso consiste en la creación de un nuevo proyecto siguiendo los siguientes pasos:

1. Escogemos *File > New Project* o directamente el botón de *New Project* en la barra de herramientas.
2. Seleccionamos la carpeta *Java > Java Application* (como cuando desarrollamos mediante código).



3. Rellenamos los datos que queramos para la clase principal.

4. Ahora tenemos que crear el archivo que contendrá la GUI. Para ello pulsamos con el botón derecho sobre el icono del proyecto que hemos creado (o en el paquete donde queremos que se ubique) y elegimos *"New > JFrame* Form".

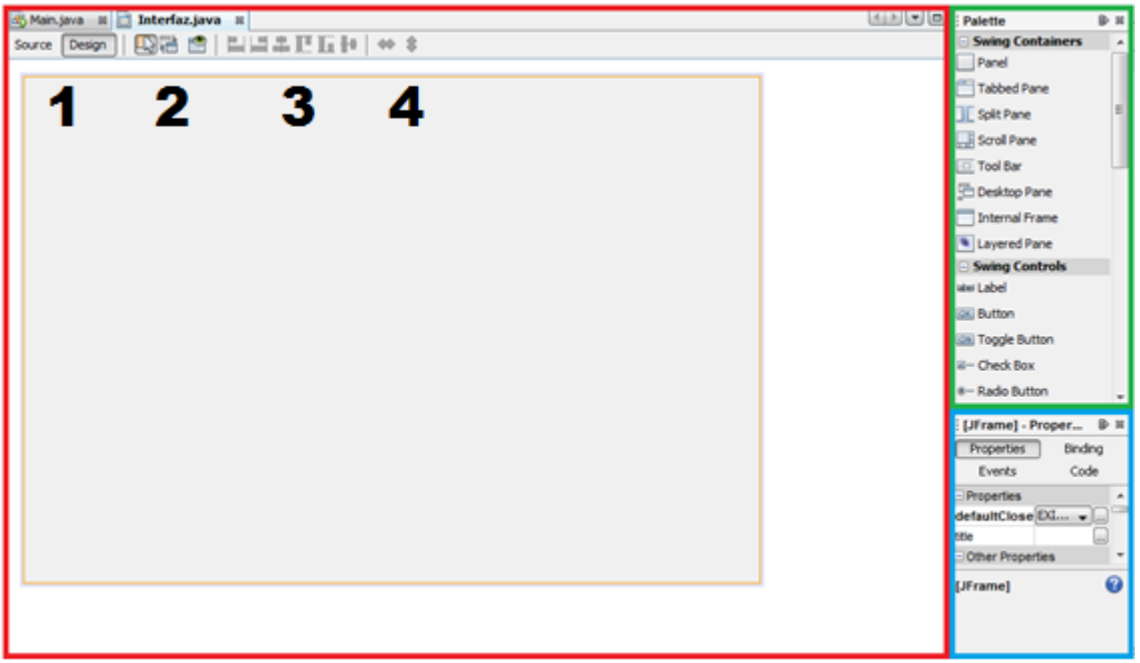


5. Rellenamos el formulario dándole el nombre que queramos al frame, por ejemplo "Interfaz".

Se nos debería abrir el archivo que hemos creado, mostrando la interfaz del GUI Builder.

Interfaz de GUI Builder

La interfaz de GUI Builder está formada por los siguientes elementos:



Zona de diseño

Esta es la zona que está contenida en el cuadrado rojo y que en un comienzo tan sólo contendrá un rectángulo vacío. Sirve para colocar los distintos elementos de la interfaz que estemos creando, cambiando sus parámetros (color, tamaño, etc.) y, en definitiva, amoldándolos a los que queramos conseguir. En la parte superior tiene una barra de herramientas con diferentes elementos (marcados del 1 al 4):

1. ***Source/Design*:** estos dos botones nos permiten cambiar entre modo de vista de código y gráfico. En el modo gráfico podremos añadir los distintos elementos de la interfaz, mientras que en el de vista de código les daremos funcionalidad. Nótese que en la parte de código aparecerán zonas sombreadas de azul, lo que significará que ese trozo de código sólo puede debería ser editado mediante el GUI Builder y no de forma manual.

```
    // Variables declaration - do not modify
    // End of variables declaration
```

2. En esta parte tenemos 3 botones muy útiles:
   - *Modo Selección*: nos permite seleccionar los elementos de la interfaz y moverlos o cambiar su tamaño.
   - *Modo Conexión*: nos deja definir la relación entre dos elementos de la interfaz, sin tener que entrar en la vista de código.
   - *Vista previa*: aparece una interfaz preliminar para que podamos evaluar su funcionalidad antes de compilar.
3. **Botones de autoajuste:** con ellos podemos alinear los elementos de la interfaz de forma automática.
4. **Pautas de auto-redimensionamiento**: indican si al ampliar la ventana principal de la interfaz los elementos que contenga se redimensionan con ella o no.

Paleta

Es la parte contenida en el rectángulo verde. En ella podremos elegir qué nuevo elemento queremos añadir al diseño de la interfaz y colocarlo según nuestras preferencias. Contiene varios apartados diferenciados:

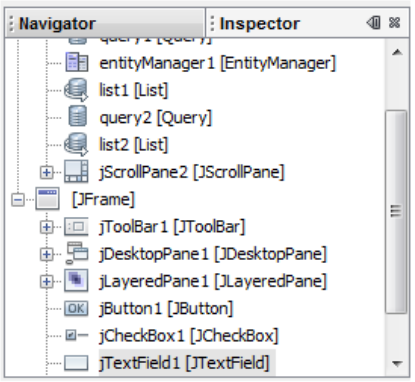
- **Swing Containers**: en esta parte se encuentran los elementos que sólo sirven para contener a otros, pero que por sí mismos no hacen nada. Ejemplos de ello son las barras de herramientas o los menús tabulares. Si queremos que contengan algo, debemos añadirlo en su interior.
- **Swing Controls:** aquí se almacenan los elementos mediante los que se crea o almacena información y órdenes, y que pueden estar contenidos en los descritos anteriormente o no. Como ejemplos, tenemos botones, etiquetas, barras deslizantes o tablas.
- **Swing Menus**: aquí hay distintos elementos que ayudan a la creación de barras de menús, añadiendo los menús propiamente dichos y sus elementos.
- **Swing Windows**: en esta sección tenemos una serie de ventanas que aparecen como diálogos, y que son útiles a la hora de crear eventos de advertencia y similares.
- **Swing Fillers**: Permiten rellenar elementos.
- **AWT**: Elementos de AWT.
- **Beans**: Permite la creación de componentes reutilizables.
- **Java Persistence**: en este último apartado se hallan los elementos correspondientes a consultas sobre bases de datos.

Propiedades de elementos

Es la parte que está en el recuadro azul. Aquí se mostrarán las propiedades del elemento que tengamos seleccionado en ese momento, permitiendo a su vez su alteración.

Inspector

Además de las zonas anteriores, existe una zona más, en la esquina inferior izquierda, llamada **Inspector**. Es aquí donde encontramos un resumen en forma de árbol de todos los elementos que hemos ido añadiendo a la interfaz, pudiendo además seleccionarlos directamente o cambiar sus propiedades pinchando en opción adecuada al hacer clic con el botón derecho del ratón sobre ellos, sin necesidad de buscarlos en la propia interfaz.

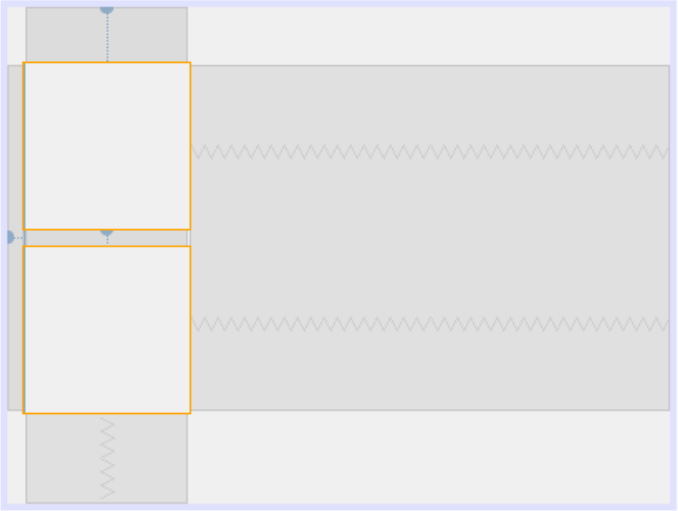


Nociones básicas de diseño

A la hora de diseñar en GUI Builder, hay 3 pasos que son claves para dar forma a la interfaz que se desea crear: añadir, modificar propiedades y definir el anclaje.

Añadir un elemento

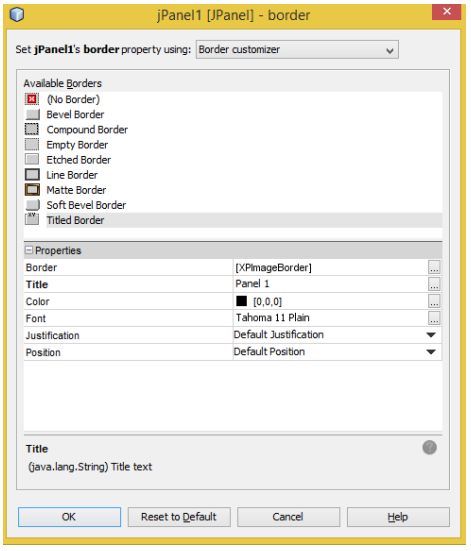
Basta con seleccionar el que queramos y arrástralo hasta la zona de diseño. En un principio suele ser mejor insertar primero los elementos contenedores, tales como paneles o menús, para luego ir añadiendo los demás. En el ejemplo hemos añadido 2 paneles:



NOTA: En la captura se ven los dos paneles porque están seleccionados, pero de no ser así no veríamos nada, pues los paneles por defecto no traen borde.

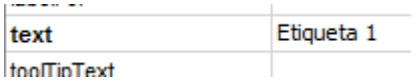
Modificar atributos

Lo primero que debemos hacer es redimensionarlos hasta estar conformes con el resultado. Después, seleccionamos el que queramos y en la zona de propiedades los modificamos a nuestro gusto. En este caso, hemos añadido un borde con título a los paneles, modificando la propiedad *Border*, escogiendo *Titled Border* y especificando el atributo de título del mismo:

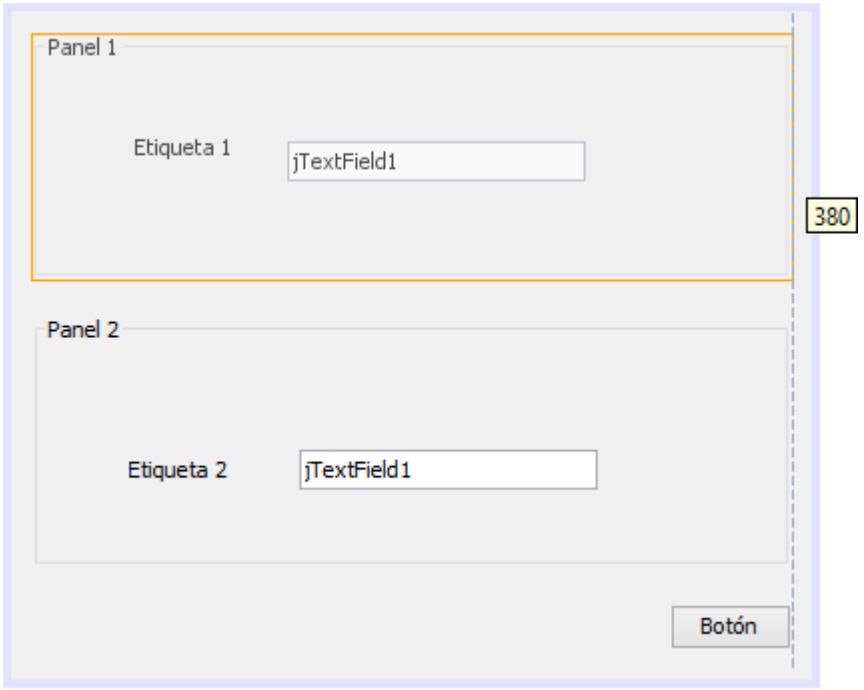


Podríamos cambiar fácilmente otros elementos como el tamaño, color, fuente…

Además hemos añadido un campo de texto y una etiqueta a cada uno de los paneles. Únicamente hemos cambiado la propiedad *text* de las etiquetas:



La ventana queda con el siguiente aspecto:

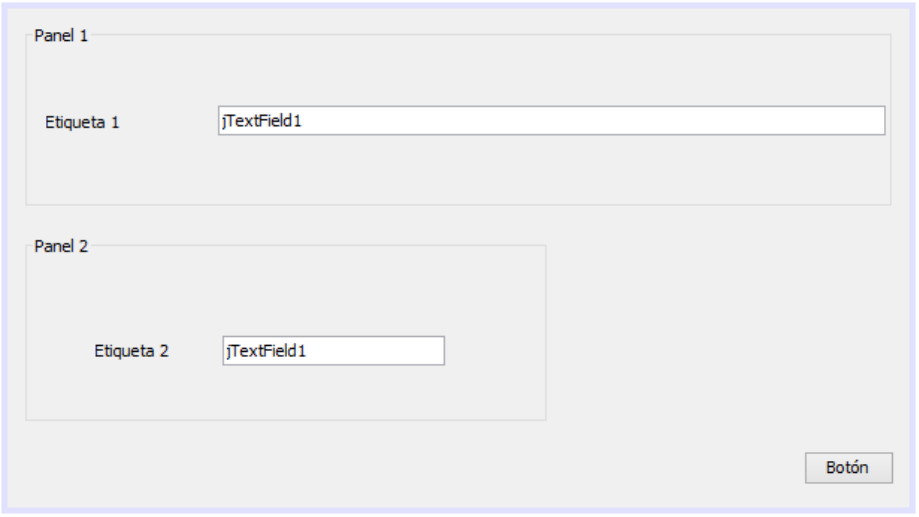


Para llevar a cabo su colocación son muy útiles las prácticas líneas que GUI Builder nos muestra para ayudarnos a conocer la distancia entre elementos. Pero estas líneas tienen otra utilidad: relacionar los elementos con el tamaño de la ventana de forma automática, lo que nos lleva al siguiente punto.

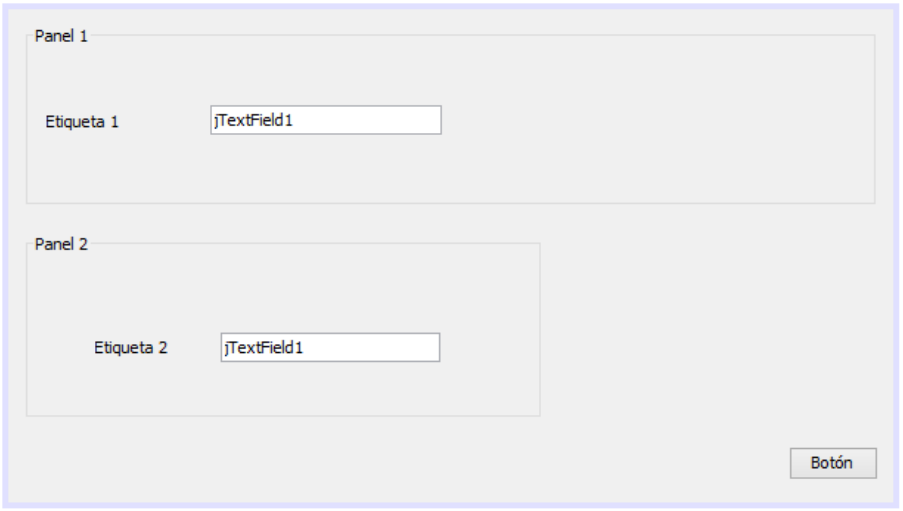
Anclaje

EL anclaje se refiere a la capacidad de un elemento de adaptarse al tamaño de la ventana o del objeto que la contiene (algo así como una interfaz dinámica en cuanto a tamaño). Se puede realizar de forma automática si acercamos un el elemento hasta el borde de la tabla de diseño o del contenedor, cuando aparece una línea punteada (imagen anterior). Para hacerlo de forma manual simplemente basta con seleccionar o deseleccionar las opciones que vimos en la zona de diseño número 4, y que relacionan contenedor con contenido en ancho, alto, o ambos a la vez.

Véase la diferencia en la siguiente imagen:



Se puede comprobar cómo el Panel 1 tiene definida la propiedad de auto redimensionarse con el ancho de la ventana, mientras que el Panel 2 queda intacto sea cual sea. Del mismo modo, `jTextField1` está relacionado con Panel 1, puesto que si no lo estuviera ocurriría lo que se ve en la siguiente imagen.



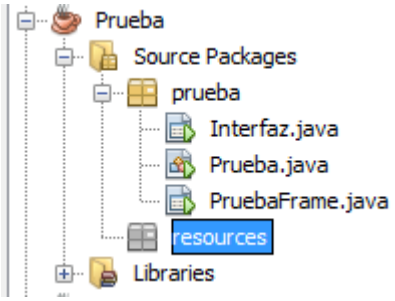
Otro tipo de anclaje es el que se da entre un elemento y otro al mismo nivel (ninguno contiene al otro), que hace que se conserve la posición relativa entre ambos (como pasa con el Botón del ejemplo, que conserva la posición con respecto al borde). Esto se consigue usando las guías que aparecen mientras se arrastra un objeto: si lo ponemos a la distancia de una de las líneas que aparecen en otro elemento, esa distancia no se alterará nunca:

Ejemplo: Añadir imágenes a un panel

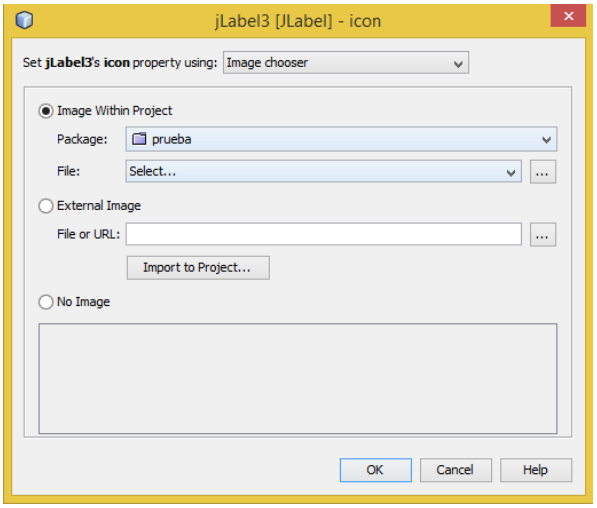
La opción de añadir una imagen en sí misma no está presente, pero se puede conseguir tras realizar una serie de pasos. Vamos a verlos a modo de ejemplo para comprender cómo puede facilitarse la tarea de implementar la interfaz:

1. Añadimos una etiqueta al proyecto (aunque parezca raro, esta etiqueta será la imagen):

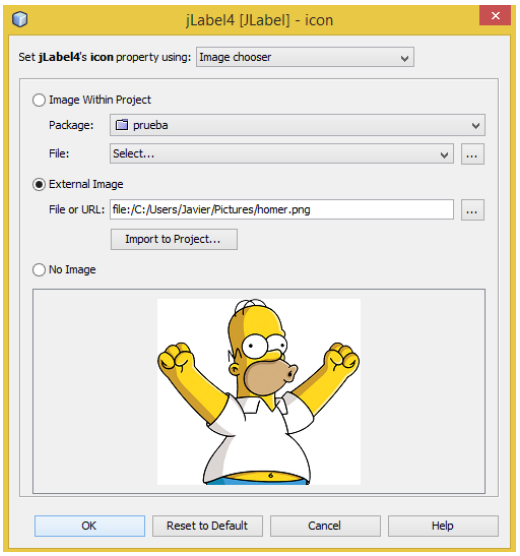
2. Creamos un paquete específico para guardar las imágenes



3. Seleccionamos la etiqueta que pusimos y elegimos "*Properties*" de las opciones que aparecen al pulsar con el botón secundario del ratón. Es un modo alternativo de acceder al panel de propiedades. Dentro de las mismas, le damos a la parte "…" del la propiedad "*icon*":



4. Seleccionamos *Import to Project*, en donde buscamos la imagen que deseamos y después en donde queremos guardarla:



Tras este paso aparecerá la imagen en la etiqueta:

5. Sólo queda modificar el nombre de la etiqueta para que éste no aparezca, si es lo que deseamos.

Ya tenemos una imagen dentro del panel, la cual quedará guardada en el .jar resultante.

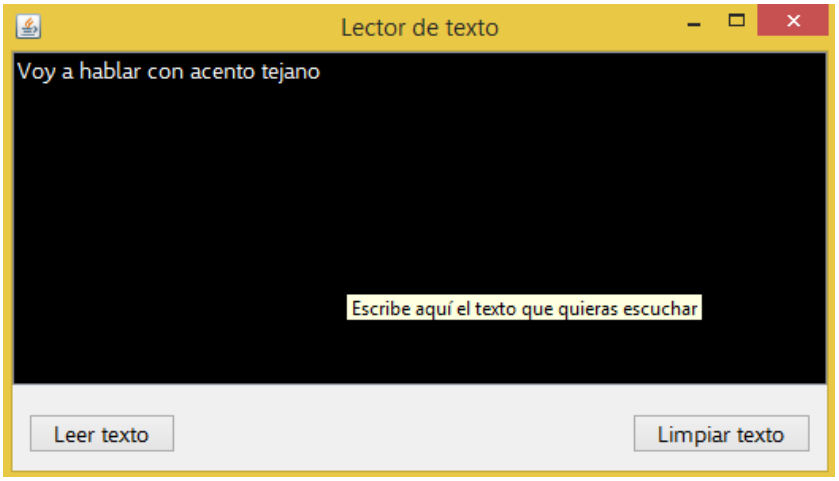
Nótese que no podemos redimensionar la imagen de fondo mediante el GUI Builder (si bien es posible mediante código).

Aplicación: Lector de texto

En lugar de ir explicando componente a componente, vamos a implementar una aplicación de ejemplo que permita poner en práctica los conceptos básicos. Con un poco de investigación es fácil averiguar cómo podemos manejar el resto de componentes, máxime habiendo trabajado con ellos en las actividades anteriores.

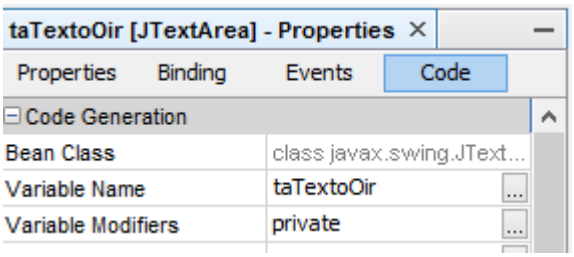
En este caso vamos a implementar una aplicación que sea capaz de leer un texto que toma un texto de un área de texto y al pulsar un botón lea el texto de la misma. Conseguiremos esto mediante una librería llamada FreeTTS (*Text To Speech)*.

La interfaz tiene el siguiente aspecto:



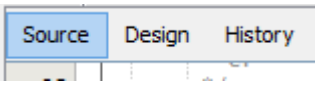
Hemos creado un frame con las siguientes características:

- Se ha cambiado un nuevo *JFrame Form* (al igual que en el ejemplo anterior) donde hemos cambiado el atributo *title*.
- Se ha añadido un elemento *Text Area* cuyos bordes se han asociado a los bordes derecho, izquierdo y superior del contenedor. Se han cambiado algunas propiedades a modo de ejemplo (*tooltip, font, foreground y background*). También cambiamos el nombre de la variable desde la pestaña *Code* del panel de propiedades para que siga la notación que usábamos hasta ahora (`taTextoOir`).



- Se han añadido dos botones *Leer texto* y *Limpiar texto* que se han asociado a los bordes izquierdo y derecho del frame respectivamente (dejando una distancia). Del mismo modo cambiaremos el texto, la fuente, el tooltip y el nombre de las variables (`btnLeer` y `btnLimpiar`).

Vamos a comprobar el texto que se ha generado desde la pestaña *Source*:



Como vemos, en principio parte del código aparece oculto. Lo que podemos ver es que se declaran las variables que hemos definido previamente:

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

    /**
     * Creates new form Interfaz
     */
    public Interfaz() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    Generated Code

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        Look and feel setting code (optional)

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Interfaz().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton btnLeer;
    private javax.swing.JButton btnLimpiar;
    private javax.swing.JButton jButton2;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTextArea taTextoOir;
    // End of variables declaration
}
```

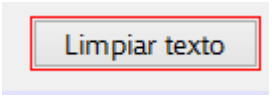
Si comprobamos el código generado (pestaña *Generated Code*) veremos el método `initComponents()` que establece las propiedades de los elementos de la interfaz:

```
private void initComponents() {

    jButton2 = new javax.swing.JButton();
    btnLimpiar = new javax.swing.JButton();
    jScrollPane1 = new javax.swing.JScrollPane();
    taTextoOir = new javax.swing.JTextArea();
    btnLeer = new javax.swing.JButton();

    jButton2.setText("jButton2");

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    setTitle("Lector de texto");

    btnLimpiar.setFont(new java.awt.Font("Kootenay", 0, 14)); // NOI18N
    btnLimpiar.setText("Limpiar texto");

    taTextoOir.setBackground(new java.awt.Color(0, 0, 0));
    taTextoOir.setColumns(20);
    taTextoOir.setFont(new java.awt.Font("Kootenay", 0, 14)); // NOI18N
    taTextoOir.setForeground(new java.awt.Color(255, 255, 255));
    taTextoOir.setRows(5);
    taTextoOir.setToolTipText("Escribe aquí el texto que quieras escuchar")
    jScrollPane1.setViewportView(taTextoOir);
```

El siguiente paso consiste en añadir código de **manejo de eventos** asociado a los botones. Hay varias maneras de hacer esto. La primera consiste en pulsar el botón *Connection Mode* que  vimos anteriormente en la barra de herramientas del modo diseño.

Tras pulsar el botón posteriormente deberemos pulsar posteriormente dos elementos:

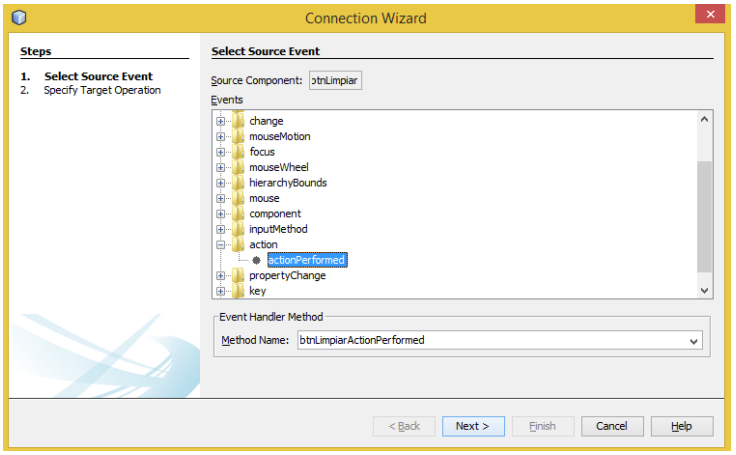
- En primer lugar pulsaremos sobre el **componente fuente** que va a desencadenar el evento. Dicho componente se mostrará rodeado con un círculo rojo. Por ejemplo, si pulsamos en el botón *Connection Mode* y posteriormente escogemos el botón *Limpiar* se mostrará el botón con el siguiente código:



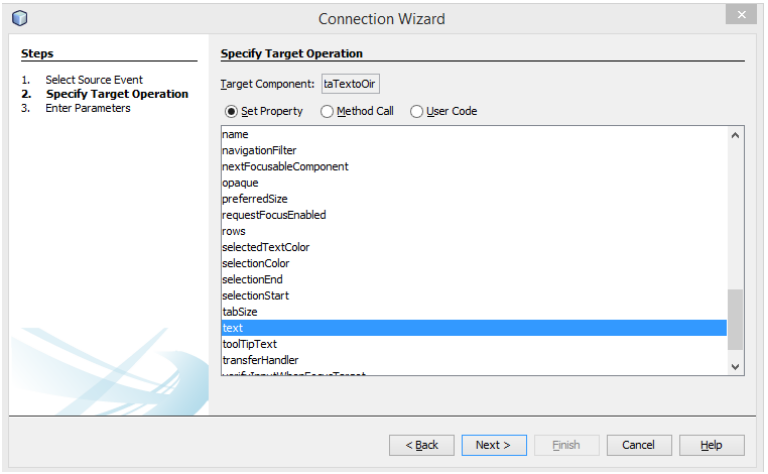
- En segundo lugar pulsamos el **componente destino** sobre el cual va a actuar el evento (en nuestro caso el área de texto). Una vez pulsado sobre el mismo se abrirá un asistente donde podemos escoger el evento que se va a tratar.

  Como vemos, al escoger el evento se genera automáticamente el nombre del método que va a tratar dicho evento. Podemos cambiar este nombre, si bien no es conveniente.

  En la siguiente pantalla hemos escogido el evento `actionPerformed` (dentro del grupo `Event`). El nombre del método que se generará será `btnLimpiarActionPerformed` (se concatena el componente origen del evento con el nombre del mismo.

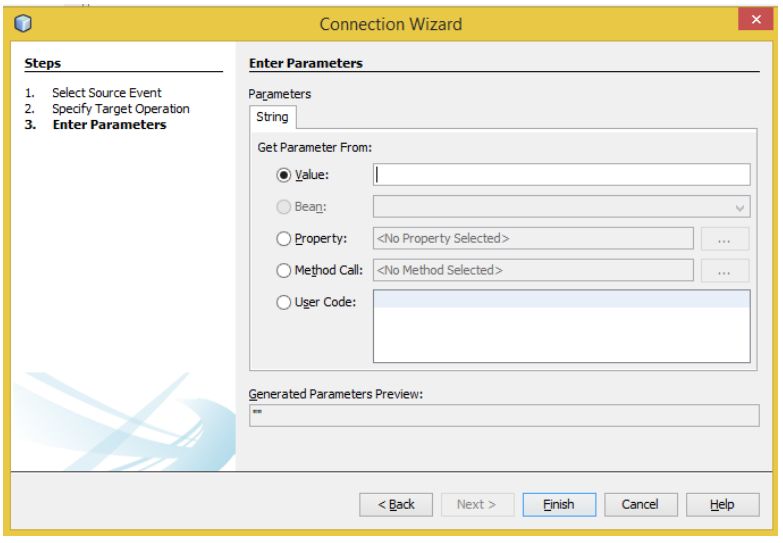


Para facilitarnos la tarea, el asistente nos sugiere acciones que podemos realizar con el componente destino. Por ejemplo, en nuestro caso vamos a limpiar el texto del área, de modo que seleccionaremos la propiedad **`text`:**



En lugar de establecer el valor de una propiedad, podríamos haber llamado a una función (que definiríamos) o meter código.

La última pantalla del asistente nos permite establecer el valor que vamos a darle a la propiedad. En nuestro caso, estableceremos la propiedad `value` del texto, y estableceremos el valor por defecto (las comillas vacías que implican un texto vacío).



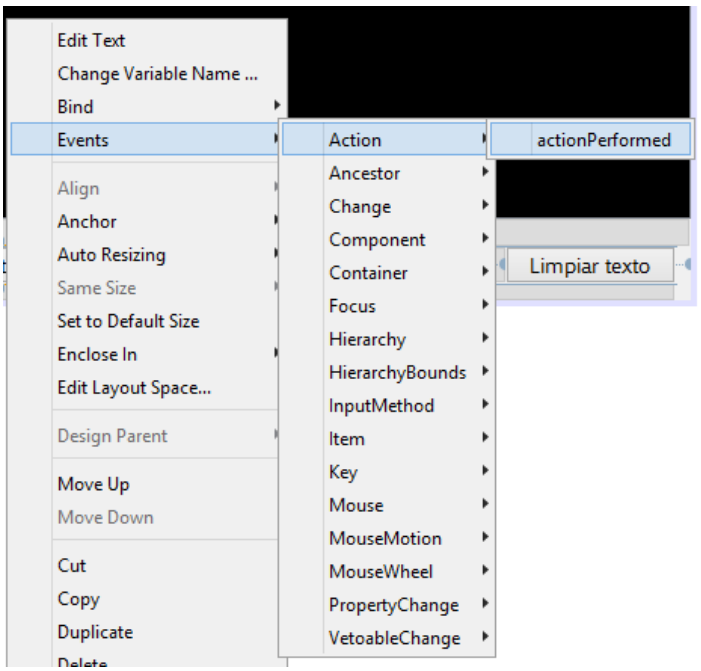
Podemos ver que se genera el código siguiente correspondiente al evento:

```
private void btnLimpiarActionPerformed(java.awt.event.ActionEvent evt) {
    taTextoOir.setText("");
}
```

NOTA: Si observamos el código autogenerado, veremos que se ha creado una clase anónima que atiende el evento e invoca al método que hemos generado desde el asistente:

```
    btnLimpiar.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            btnLimpiarActionPerformed(evt);
        }
    });
```

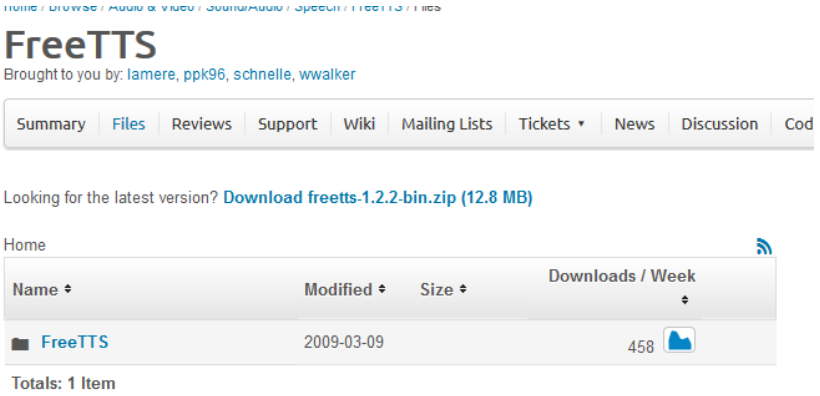
Otra manera de añadir eventos (más adecuada si no tenemos un componente destino) es pulsar con el botón derecho sobre el componente que genera el evento y escoger la opción *Eventos* del menú contextual, navegando a continuación por el propio menú para escoger el evento que queremos tratar:



Veremos que se abre el editor de código donde podemos añadir el código a la función que tratará el evento:

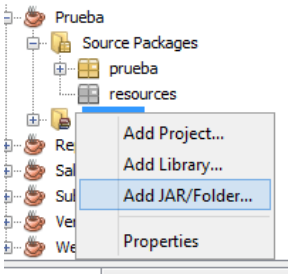
```
private void btnLeerActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}
```

Aquí añadiremos el código necesario para transformar el texto a voz. Usaremos la librería FreeTTS, que podemos descargar desde la página http://freetts.sourceforge.net/docs/index.php, pulsando la sección *Downloading and Installing* y a continuación el botón de *Download*. Esto nos llevará a la página sourceforge, desde donde podemos descargar un zip con la última versión:

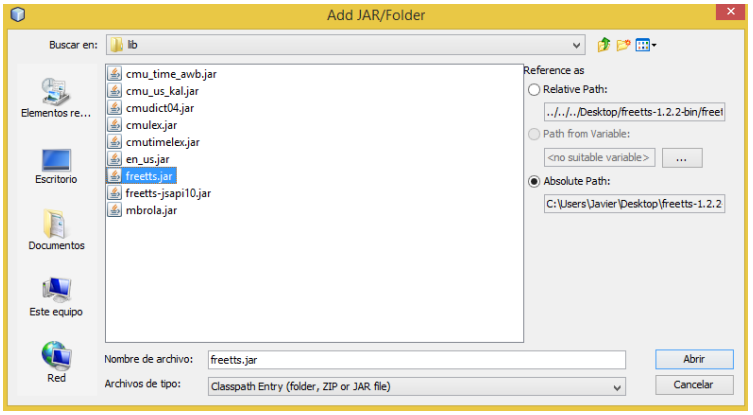


Una vez descargado el zip lo descomprimimos en un directorio cualquiera de nuestro ordenador.

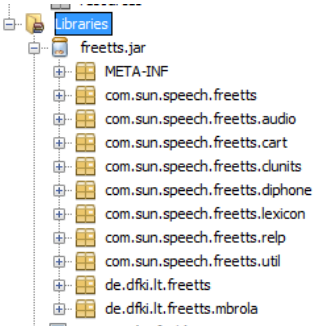
El siguiente paso consiste en añadir la librería a nuestro proyecto. Para ello pulsamos con el botón derecho sobre la carpeta librerías del mismo y escogemos *Add JAR/Folder*:



Se abrirá un árbol de navegación. Debemos ir al directorio donde hemos descomprimido FreeTTS, y dentro del mismo en el subdirectorio `lib`, para escoger el archivo `freetts.jar`:



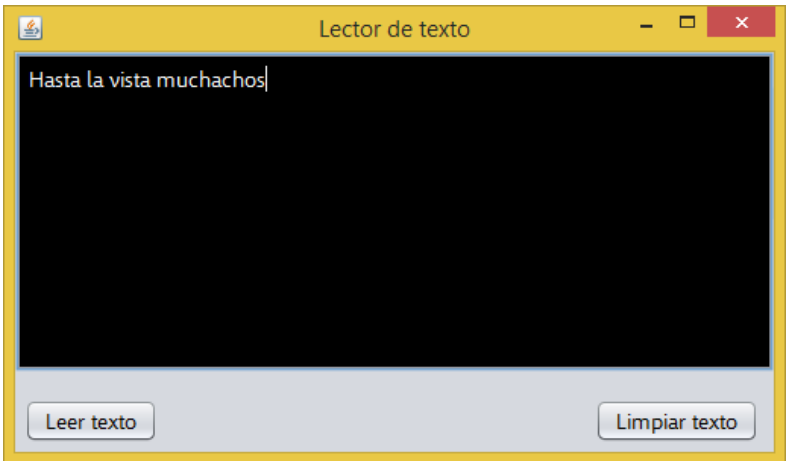
Veremos que el jar se ha añadido a la carpeta de librerías, y junto con él las clases que forman parte de la librería:



Ya podemos escribir el código correspondiente a la gestión del evento, que se ha construido a partir de la documentación de la librería. Para más información podemos consultar la misma:

```
private void btnLeerActionPerformed(java.awt.event.ActionEvent evt) {
    String texto=taTextoOir.getText();
    if(texto!=null && texto.trim().length()!=0){
    VoiceManager gestor=null;
    Voice vozSintetica=null;

    try{
        gestor=VoiceManager.getInstance();
        vozSintetica=gestor.getVoice("kevin");
        vozSintetica.allocate();
        vozSintetica.speak(texto);
    }
    catch(Exception e){
        e.printStackTrace();
    }
    finally{
        vozSintetica.deallocate();
    }
   }
}
```

Para comprobar si el programa funciona correctamente debemos ejecutarlo (no es suficiente con la vista previa):



Ventajas e inconvenientes

Lógicamente la elaboración de interfaces mediante herramientas automáticas como la que acabamos de ver tiene **ventajas**, entre las que podemos citar:

- La **curva de aprendizaje** es muy reducida. Es decir, es muy fácil comenzar a hacer aplicaciones de manera rápida.
- El **tratamiento de las propiedades** es muy cómodo, en especial en lo que se refiere a propiedades complejas como bordes, fuentes o iconos.
- El **modelo de distribución** de componentes dentro de un frame también se facilita mucho respecto a la implementación manual.
- También se facilita la elaboración de **componentes laboriosos** como pueden ser los menús o las barras de herramientas.

Sin embargo el desarrollo de interfaces con herramientas de desarrollo rápido también tiene ciertas **desventajas**:

- El código que se va generando resulta bastante **difícil de mantener**, en especial si no tenemos cuidado de ir nombrando y redistribuyendo adecuadamente los elementos que se generan de acuerdo a nuestro criterio.
- Es necesario haber comprendido previamente algunos de los **conceptos** para poder aplicarlos. Por ejemplo, la parte de la gestión de eventos.
- El código también es muy **dependiente de la herramienta** con la cual lo generamos. Por tanto, nos "atamos" al desarrollo con una plataforma o herramienta concreta.
  Las herramientas pueden cambiar o incluso dejar de desarrollarse. Por el contrario, la tecnología que subyace (el código) es estable.

Por otro lado, aunque la curva de aprendizaje al codificar manualmente es más pronunciada al principio, tras unos meses de experiencia apenas se nota diferencia en los tiempos de desarrollo utilizando ambos enfoques.

En muchos casos se usa un **enfoque mixto**, generando parte de la interfaz con una herramienta automática y codificando el resto de la aplicación siguiendo el MVC. Sin embargo, es fundamental haber comprendido correctamente los fundamentos de funcionamiento de Swing para entender el modo en que la herramienta funciona y cómo ir corrigiendo los posibles problemas que se puedan ir presentando.