Simple Notepad

ADRIÁN BARRIO ANDRÉS

JAVA.SWING

Contenido

Estructura de la aplicación

Dada la necesidad de implementar un patrón MVC en Swing me he visto obligado a crear una esctructura de paquetes interna que detallo a continuación:

- **assets** - Contiene todos los recursos gráficos necesarios para las vistas de la aplicación, iconos y demás, la libería de iconos utilizada es la Tango Icon Library de uso libre.
- **controllers** - Contiene el controlador principal de la aplicación, que actua sobre la vista.
- **helpers** – Contiene los "ayudantes" o manejadores del contenido que se han necesitado para llevar a cabo la aplicación.
- **lib** – Contiene librerías externas utilizadas.
- **models** – Contiene el modelo de datos con el retorno de los datos necesarios que el controlador necesita.
- **styles** – Contiene los estilos aplicados aparte a la vista, tales como cambiar el fondo de los botones y su comportamiento cuando pasa el ratón por encima.
- **views** – Contiene el diseño de las vistas con las que el usuario interactua.

Funcionamiento

La aplicación arranca en el **Main.java** donde se invoca al modelo y la vista (EditorModel y EditorView) y se pasan al controlador (EditorController) para que él se encargue de realizar la lógica.

En el constructor del controlador se dan de alta variables privadas referenciadas a las clases del modelo y la vista y se empieza a trabajar con ellos en el método **init()**.

Github

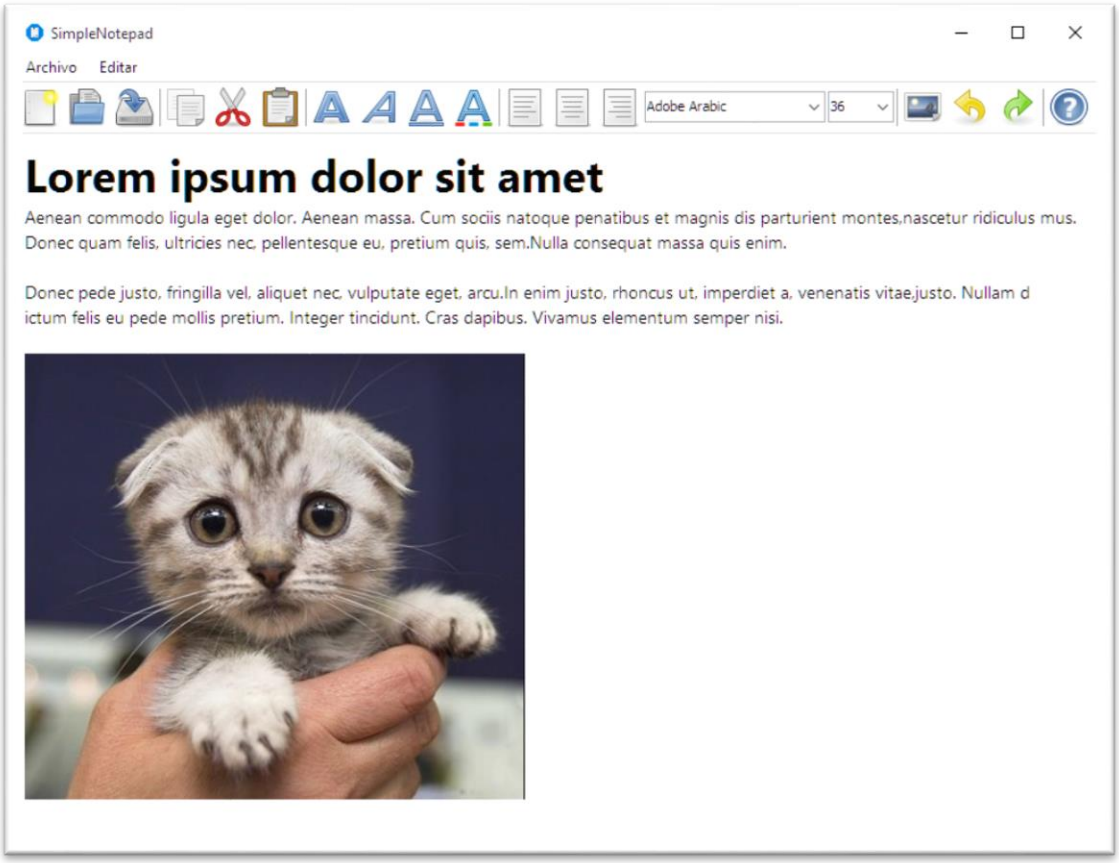
Para facilitar el acceso al proyecto lo he subido a Github, el cuál seguiré actualizando cuando tenga tiempo.

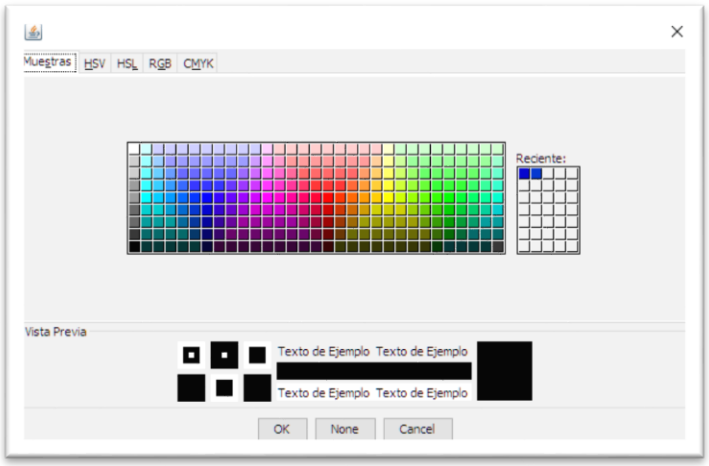
https://github.com/statickidz/SimpleNotepad-Swing

Características

- Posibilidad de aplicar estilos al texto seleccionado
- Crear documentos
- Guardar documentos en formatos DOC, HTML y SNP (extensión propia)
- Auto añadido de extensión al guardar documentos y aviso de sobre escritura
- Posibilidad de imprimir documentos (ergo guardar como PDF)
- Copiar, cortar y pegar. Desde el botón derecho, con los métodos abreviados del teclado y desde el menú
- Formateo de texto: negrita, itálica, subrayado
- Cambio de color mediante un selector
- Alineación de texto: izquierda, centro y derecha
- Cambio de fuente, eligiendo las del sistema y tamaño de la misma
- Inserción de imágenes: Se ha empleado el formato Base64, muy utilizado en HTML para convertir las mismas con una librería externa (no se ha implementado para Word, por lo que no se pueden visualizar, habría que convertirlas a RTF o a XML formateado para que este las admita)
- Deshacer y rehacer, mediante teclado y botones

Capturas


SimpleNotepad
Archivo
Editar
Adobe Arabic
36
Lorem ipsum dolor sit amet
Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes,nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem.Nulla consequat massa quis enim.
Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu.In enim justo, rhoncus ut, imperdiet a, venenatis vitae,justo. Nullam d ictum felis eu pede mollis pretium. Integer tincidunt. Cras dapibus. Vivamus elementum semper nisi.


Muestras
HSV
HSL
RGB
CMYK
Reciente:
Vista Previa
Texto de Ejemplo Texto de Ejemplo
Texto de Ejemplo Texto de Ejemplo
OK
None
Cancel

Snippets útiles

Añadir soporte para click derecho

```
view.textPane.addMouseListener(textPaneMouseListener());

private MouseListener textPaneMouseListener() {
     return new MouseAdapter() {
         @Override
         public void mouseReleased(final MouseEvent e) {
             if (e.isPopupTrigger()) {
                 JPopupMenu menu = new JPopupMenu();
                 JMenuItem item;

                 item = new JMenuItem(new DefaultEditorKit.CopyAction());
                 item.setText("Copiar");
                 menu.add(item);

                 item = new JMenuItem(new DefaultEditorKit.CutAction());
                 item.setText("Cortar");
                 menu.add(item);

                 item = new JMenuItem(new DefaultEditorKit.PasteAction());
                 item.setText("Pegar");
                 menu.add(item);

                 menu.show(e.getComponent(), e.getX(), e.getY());
             }
         }
     };
 }
```

Imprimir documento

```
private void printDocument() {
     try {
         boolean done = view.textPane.print();
         if (done) {
             JOptionPane.showMessageDialog(null, "Impresión correcta");
         } else {
             JOptionPane.showMessageDialog(null, "Error al imprimir");
         }
     } catch (PrinterException | HeadlessException pex) {
         JOptionPane.showMessageDialog(null, "Hubo un error al imprimir");
     }
 }
```

Insertar una imagen

```
private void insertImage() {
     JFileChooser fc = new JFileChooser();
     FileNameExtensionFilter pngFilter =
           new FileNameExtensionFilter("PNG (*.png)", "png");
     fc.setFileFilter(pngFilter);
     FileNameExtensionFilter jpgFile =
           new FileNameExtensionFilter("JPEG (*.jpg)", "jpg");
     fc.setFileFilter(jpgFile);
     fc.setFileSelectionMode(JFileChooser.FILES_AND_DIRECTORIES);
     if (fc.showDialog(view, "Insertar") != JFileChooser.APPROVE_OPTION) return;
     String filename = fc.getSelectedFile().getAbsolutePath();
     if (filename == null) return;
     try {
         String encoded = Base64.encodeFromFile(filename);
         String imgHtml = "<img src=\"" + encoded + "\">";
         try {
             customHtmlEditorKit.insertHTML(
                  document,
                  view.textPane.getSelectionStart(),
                  imgHtml, 0, 0, null);
         } catch(BadLocationException e) {
             System.out.println("Bad insert image");
         }

     } catch (IOException e) {
         JOptionPane.showMessageDialog(view, "Could not find file: " + filename);
     }
 }
```