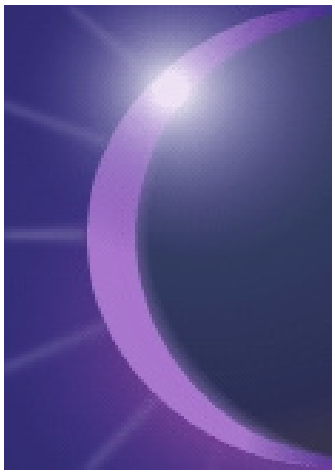
Eclipse
Visual Editor



Universidad Nacional de la Patagonia Austral

Unidad Académica Río Gallegos

Analista de Sistemas – Licenciatura en Sistemas

Laboratorio de Programación

Lic. Verónica L. Vanoli

Indice

Introducción

Eclipse es un entorno de desarrollo integrado de código abierto con licencia libre para programar en Lenguaje Java que no contiene entorno gráfico de Eclipse, disponible en la web oficial www.eclipse.org. Para ello, es necesario instalarle un plugin adecuado, entre los más conocidos se encuentran: Eclipse Visual Editor, SWT GUI Builder, Java Advanced Imaging, Advanced Eclipse SWT Designer, JLense, Jelly SWT, Workzen, Sweet-SWT, JBeaver Swing GUI Builder, Jigloo, entre otros.

La base para Eclipse es la Plataforma de Cliente Enriquecido (Rich Client Platform - RCP). Los siguientes componentes constituyen la plataforma de cliente enriquecido:

- Plataforma principal - inicio de Eclipse, ejecución de plugins.
- OSGi - una plataforma para bundling estándar.
- El Standard Widget Toolkit (SWT) - Un widget toolkit portable.
- JFace - manejo de archivos, manejo de texto, editores de texto.
- El Workbench de Eclipse - vistas, editores, perspectivas, asistentes.

Los widgets de Eclipse se encuentran implementados por una herramienta de widget para Java llamada SWT, a diferencia de la mayoría de las aplicaciones Java, que usan las opciones estándar Abstract Window Toolkit (AWT) o Swing. La interfaz de usuario de Eclipse también tiene una capa GUI intermedia llamada JFace, la cual simplifica la construcción de aplicaciones basada en SWT.

En cuanto a las aplicaciones clientes, eclipse provee al programador frameworks enriquecidos para el desarrollo de aplicaciones gráficas, definición y manipulación de modelos de software, aplicaciones web, etc. Por ejemplo, GEF (Graphic Editing Framework) es un plugin de Eclipse para el desarrollo de editores visuales a partir de modelos de aplicaciones, que pueden ir desde procesadores de texto WYSIWYG hasta editores de diagramas UML, diagramas de clases/flujos/actividad/entidad-relación, máquinas de estado, interfaces gráficas para el usuario (GUI), etc. Dado que los editores realizados con GEF “viven” dentro de Eclipse, además de poder ser usados conjuntamente con otros plugins, hacen uso de su interfaz gráfica personalizable y profesional. Acompaña a este plugin el widget Draw2D que se encarga de gestionar todo lo relacionado al rendering en pantalla. Consiste de un compuesto SWT, un sistema liviano y sus contenidos: figuras.

La Fig. 1, representa lo mencionado en estos dos últimos párrafos.

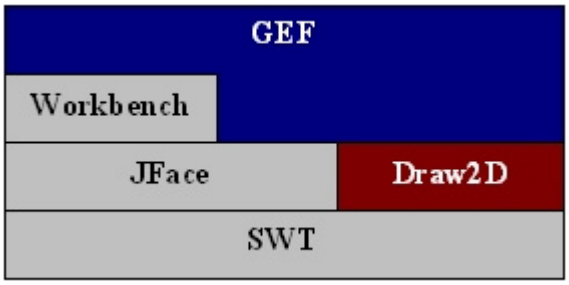


Fig. 1: Capas de Eclipse.

Eclipse Visual Editor (VE) es un entorno de creación de interfaces gráficos de usuario en Eclipse, muy fácil de usar, dirigido no solamente a Swing/JFC y SWT, sino también a otros lenguajes (como C/C++) y widgets. Dispone de editores gráficos avanzados, soporta Windows y Linux/GTK,

permite editar tanto el código Java como el modo visual y que se reflejen los cambios. VE necesita de las siguientes aplicaciones para funcionar: principalmente del Eclipse, el Eclipse Modeling Framework (EMF) y el GEF.

EMF es una estructura Java/XML para generar herramientas y otras aplicaciones a partir de modelos de clases simples. Como parte del mismo se encuentra relacionada la librería para modelar, usada por muchos otros proyectos, llamada Eclipse Java EMF Model (JEM).

Como instalarlo…

Off-line (vía Zip)

Paso 1. Se recomienda tener instalado el Eclipse 3.2.2 y no debe encontrarse abierta dicha aplicación. Además, debe contarse con los siguientes paquetes: EMF 2.2.0, GEF 3.2, JEM 1.2.3 y VE 1.2.3.
Paso 2. Descomprimir cada uno de los paquetes en la misma carpeta en que se encuentra el eclipse instalado (sobrescribiendo los archivos existentes).
Paso 3. Una vez completado el paso 2, ahora el VE se encuentra listo para su uso.
Paso 4. Abrir el eclipse.

Nota: las versiones nombradas anteriormente son recomendaciones para el correcto uso del VE. Una vez instalado es actualizable a futuro.

On-line (vía Update)

Paso 1. Se recomienda tener instalado el Eclipse 3.2.2 y debe encontrarse abierto, para instalar el VE desde el update de Eclipse.
Paso 2. Abrir el actualizador de eclipse desde el menú principal: Help → Software Updates → Find and Install…
Paso 3. Seleccionar “Search for new features to install” y click al botón Next.
Paso 4. Click al botón “New Remote Site…”. Introducir en Name: Visual Editor y en URL: http://update.eclipse.org/tools/ve/updates/1.0. Click al botón OK.
Paso 5. Repetir el paso anterior, pero esta vez, introducir en Name: Eclipse Modeling Framework y en URL: http://update.eclipse.org/tools/emf/updates. Click al botón OK.
Paso 6. Repetir el paso anterior nuevamente e introducir en Name: Old Eclipse y en URL: http://update.eclipse.org/updates/3.1. Click al botón OK.
Paso 7. Una vez agregados los nuevos sitios, marcarlos con un tilde. Click al botón Finish.
Paso 8. Seleccionar un sitio de donde bajar el paquete (mirror), si es posible el más cercano. También en el paso 7, se puede activar la opción de seleccionar un mirror automáticamente.
Paso 9. Expandir el árbol Visual Editor – VE y seleccionar Visual Editor SDK 1.2.1.
Paso 10. Luego, lo mismo para Eclipse Modeling Framework – EMF SDK 2.2.1 y seleccionar EMF SDK 2.2.1.
Paso 11. Igual para Old Eclipse – GEF 3.1.1 y seleccionar Graphical Editing Framework 3.1.1.
Paso 12. Click al botón Next. Aceptar las licencias. Click al botón Next y luego al botón Finish.
Paso 13. Ahora el VE se encuentra instalado.

Como utilizarlo…

A partir de un nuevo proyecto o proyecto existente, para poder crear una aplicación gráfica debe agregarse una clase visual al proyecto desde el menú: File → New → Other… Seleccionar la opción "Visual Class" (puede encontrarse dentro de Java). Click al botón Next. Ingresar los datos principales de una clase (nombre, paquete, modificador de acceso, etc.) y en este caso la diferencia se encuentra en determinar el estilo a implementar para convertirlo en la interfaz gráfica que corresponda. Los estilos básicos son los contenedores de AWT/Swing. Click al botón Finish.

Veamos por pasos, como sería implementar de esta forma el ejemplo del apunte "Interfaz Gráfica de Usuario en Java" de la página 21. Vamos a adaptarlo a Swing para mejorar el aspecto del mismo.

Ejemplo: Clase Visual VentanaDibujo.java

Una vez creada la clase visual VentanaDibujo, podemos apreciar que la vista de Edición se divide en tres partes (Fig. 2):
A. donde se encuentra el código correspondiente a la clase creada (en este caso VentanaDibujo). Inicialmente dicho código se crea de forma automática de acuerdo al estilo elegido (en este caso un JFrame), posteriormente puede continuar automáticamente o editable de forma manual.
B. donde se encuentra representado gráficamente el estilo elegido. Básicamente aparece el contenedor inicial (en este caso el JFrame) al cuál se le podrá modificar las características del mismo y agregar los respectivos componentes.
C. la paleta (Palette) propia del VE, para diseñar cualquier contenedor de forma más rápida.

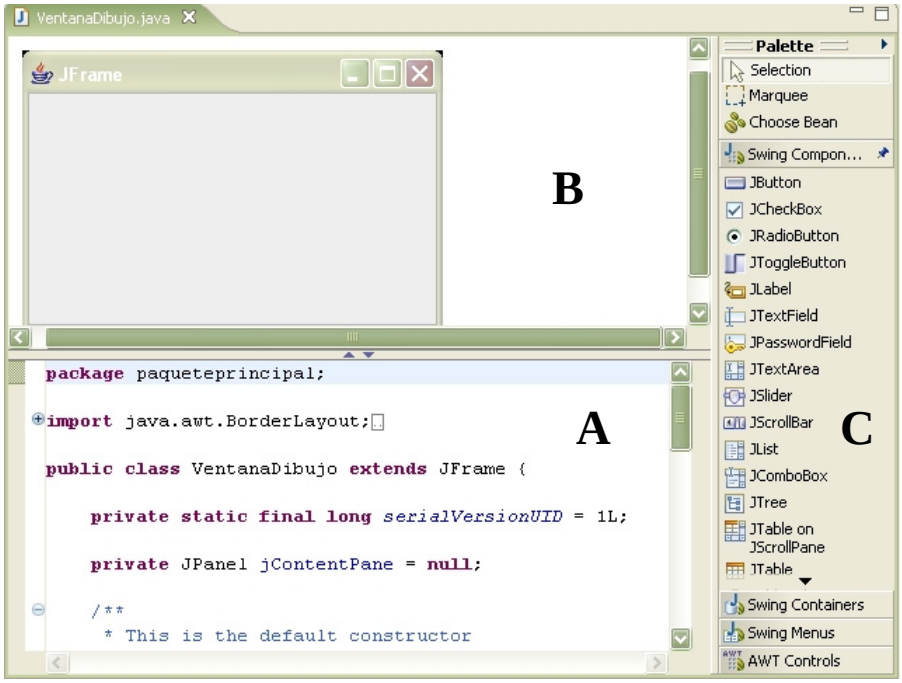


Fig. 2: Vista de Edición de Eclipse.

Algunas de las características que pueden modificarse del contenedor, en forma ágil, son: agregarle eventos (Events), cambiar el título (Set Title) y, agregar y actualizar el layout (Set Layout/Customize Layout…). Esto mismo se logra haciendo click con el botón derecho sobre el JFrame en la parte B de la vista. El código va generándose automáticamente en la parte A de la

vista. También pueden modificarse las medidas del contenedor, utilizando los ocho puntos de referencia del contorno de la figura, una vez seleccionada la misma.

A partir de ahora el JFrame se encuentra en condiciones de agregarle los componentes correspondientes. Los mismos pueden seleccionarse desde la paleta y agregar dentro del JFrame. Para este ejemplo, los componentes son: los paneles, el menú, los botones de radio, las etiquetas, los campos de texto, el botón y la lista desplegable. Cada vez que se incorpora un componente debe hacerse en una zona del JFrame permitida. En cada agregado es posible utilizar el nombre por definición de cada componente o asignarlo de forma personalizada (no será más que el nombre del atributo dentro de la clase).

Paneles

Se selecciona de la Paleta la opción Swing Containers (Contenedores de Swing) y dentro de ésta se escoge, mediante un click, el contenedor JPanel. Luego se lleva dicho contenedor al JFrame con sólo hacer click sobre el mismo. Si es el primer panel que se agrega, sólo deberá realizase en el centro del JFrame, en cambio, si ya existen paneles, deberá elegirse el panel que contendrá al nuevo agregado. Luego aparecerá la ventana para el nombre del panel (Fig. 3), donde puede dejarse el nombre del componente por definición o cambiarlo. También puede determinarse que no aparezca más dicha ventana y el nombre sea siempre generado automáticamente; con sólo activar la opción “Do not ask again”.

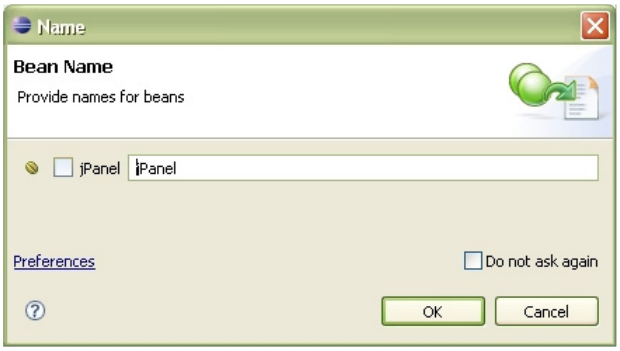


Fig. 3: Ventana para nombre de Componentes/Contenedores.

De la misma manera, van agregándose los paneles necesarios. Como antes se ha mencionado, al contenedor se le puede cambiar algunas características básicas, en este caso, modificar el Layout a cada panel agregado (con sólo hacer click en el contenedor con el botón derecho). Otras características como cambiarle el color, se realizan desde el código (escribiendo dentro del método correspondiente, por ejemplo: *jPanel2.setBackground(Color.gray)*).

Cada vez que se selecciona un componente que ya ha sido agregado, puede verse el algoritmo que corresponde al componente en el código de la clase. Esto puede servir para cualquier cambio que quiera realizarse. Para ello debe estar activada la opción Selection (Selección) de la Paleta. Si se encuentra activada la opción Macar (Marquee) de la Paleta, podrá seleccionarse un conjunto de componentes, por ejemplo para eliminarlos o para moverlos de lugar.

Menú

De la misma manera que se agregó un panel, también puede agregarse una barra de menú, seleccionando de la Paleta la opción Swing Menus (Menúes de Swing) y luego JMenuBar. Para incorporarle los diferentes menúes, se debe ir eligiendo un JMenu e incorporarlo a la barra. Puede cambiarse el texto desde el componente mismo o en el código generado. Además, cada menú podrá

contar con sus respectivos JMenuItem. La Fig. 4, muestra como ha quedado hasta el momento el JFrame.

Fig. 4: JFrame luego de agregar paneles y menú.

Botones de radio, etiquetas, campos de texto, botón y lista desplegable

Para finalizar la ventana principal de este ejemplo falta incorporar el resto de los componentes. En este caso el panel derecho será el encargado de contenerlos. Hay que tener en cuenta que dependiendo del Layout que se utilice es como se verán distribuidos los componentes.

Primero se agregan los JRadioButton, le siguen los JTextField acompañados de sus respectivos JLabel. Previo a estos últimos pares de componentes puede agregarse un panel y a él los mismos y así logran agruparse correctamente. Le siguen el JButton y por último el JComboBox, que también pueden utilizar otro panel para su distribución. Todos ellos pueden cambiar rápidamente su título haciendo click con el botón derecho (Set Text). El resultado final puede verse en la Fig. 5, donde se aprecian los distintos paneles agregados que permiten la distribución de los componentes.

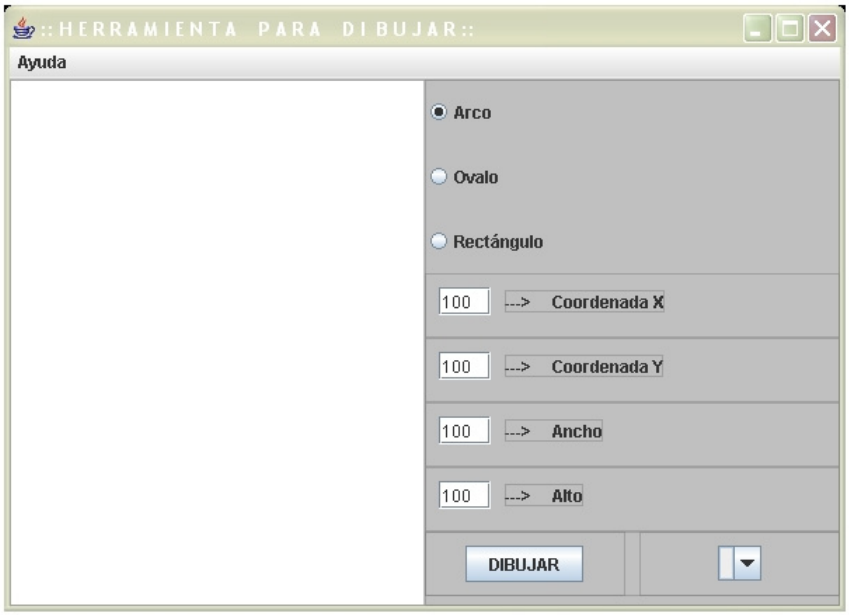


Fig. 5: JFrame luego de agregar el resto de los componentes.

Todas las operaciones realizadas sobre la paleta y las características rápidas sobre el gráfico, pueden accederse desde la barra de herramientas. El código de la clase puede verse representado como un árbol en la Vista General, etiqueta Java Beans, cuya raíz es el contenedor principal y el resto de los "nodos" son todos los componentes que posee él mismo organizados según su distribución. Lo interesante de esta vista es la facilidad de acceso a cada elemento propio de la clase visual, su organización y al hacerse click en cada uno de ellos, inmediatamente puede verse seleccionado en la parte gráfica y el código correspondiente.

Cabe recordar que debe respetarse la forma en que se implementa el código automáticamente, dado que cualquier cambio manual que no cumpla con la codificación, resultará ser un error y seguramente la representación gráfica podrá salir mal o aún peor no mostrar nada.

Ejecución de la Clase Visual

Si bien, a medida que va actualizándose la clase visual, puede ir visualizándose el resultado en pantalla, algunos componentes/contenedores pueden no estar representados en su formato definitivo. Por ejemplo, en este caso ocurre con el JComboBox. Por lo tanto, existe la posibilidad de ejecutar la clase para visualizar el resultado definitivo del contenedor que se encuentra diseñando. Para ello, desde el menú: Run → Run As → Java Bean (Fig. 6).

Fig. 6: Clase VentanaDibujo ejecutada.

Abrir una Clase Visual Existente

En el caso de utilizar un proyecto existente que contenga una clase visual que ya ha sido creada, la misma puede abrirse de la siguiente forma:

a) como un Java Editor (Editor de Java), en dicho caso se abrirá como una clase común de java donde el editor sólo visualizará el código para su actualización en la vista de edición;

b) como un Visual Editor (Editor Visual) donde volverá a aparecer la vista de edición distribuida con sus tres partes correspondientes.

Dependiendo de la complejidad del contenedor es probable que tarde unos segundos en abrirse la parte gráfica, en dicho caso aparecerá un cartel avisando que se encuentra cargando.

Eventos

Como se ha mencionado anteriormente, se le puede agregar eventos a los contenedores tan solo seleccionándolo y haciendo click con el botón derecho. Lo mismo ocurre con los componentes. Para el ejemplo anterior, puede agregarse un evento a la ventana principal (JFrame) para que la misma pueda cerrarse al hacerse click en la cruz (evento windowclosing). Automáticamente se crea el código correspondiente a dicho evento:

```
this.addWindowListener(new java.awt.event.WindowAdapter()
	{
	public void windowClosing(java.awt.event.WindowEvent e)
		{
		System.out.println("windowClosing()");
		//TODO Auto-generated Event stub windowClosing()
		}
	});
```

Lo que resta hacerse en forma manual, es agregar el código que corresponda a dicho evento. Por ejemplo:

```
this.addWindowListener(new java.awt.event.WindowAdapter()
	{
	public void windowClosing(java.awt.event.WindowEvent e)
		{
		System.out.println("Cierre de la ventana...");
		System.exit(0);
		}
	});
```

Otro evento, podría ser el de hacer click en el botón “DIBUJAR” (evento actionPerformed). En este caso el código resultante sería:

```
jButton.addActionListener(new java.awt.event.ActionListener()
	{
	public void actionPerformed(java.awt.event.ActionEvent e)
		{
		dibujar(jButton);
		}
	});
```

Y así, con todos los contenedores/componentes que requieran de un evento.

Es importante destacar que el código que genera el VE automáticamente aplica correctamente la modularidad, a cada componente agregado le corresponde un método, esto tiene asociado los beneficios de la modularización. Caso contrario ocurre con los eventos, dado que el código correspondiente a un evento de un componente se incorpora inmediatamente en el lugar donde él mismo se encuentra ubicado. En el caso de que existan muchos eventos asociados, esto se torna ineficiente. Por lo tanto, se recomienda arreglar dicho código utilizando los Escuchas vistos en la etapa de Interfaz Gráfica de Usuario en Java.

Bibliografía utilizada

- http://es.wikipedia.org/wiki/Eclipse_(software)
- http://wiki.eclipse.org/VE
- http://www.eclipsecon.org/2004/EclipseCon_2004_TechnicalTrackPresentations/47_Hudson.pdf
- Java™ 2 Platform Standard Edition 5.0 API (Application Programming Interface) Specification
- Website Eclipse: http://www.eclipse.org
- Website Eclipse - Graphic Editing Framework: http://www.eclipse.org/gef/
- Website Eclipse - Visual Editor Project: http://www.eclipse.org/vep/WebContent/main.php