

Extensión de componentes 2

José Jarones Bueno

Crear un JPanel con imagen de fondo

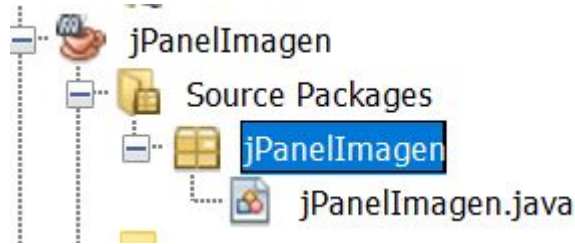
Vamos a crear nuestro propio componente a partir de componentes ya creados.

Vamos a crear un panel que permita poner una imagen de fondo en el panel, y que la imagen pueda tener una transparencia.



Creamos un proyecto para el componente

Ejemplo creamos un proyecto llamado jPanelImagen con un paquete jPanelImagen y creo la clase jPanelImagen.java



La clase JPanelImagen

Debe extends de JPanel y debe implements Serializable

```
package JPanelImagen;

import java.io.Serializable;
import javax.swing.JPanel;

/**
 *
 * @author jaron
 */
public class JPanelImagen extends JPanel implements Serializable {

}
```

Creamos private File rutaImagen y getter y setters

```
public class JPanelImagen extends JPanel implements Serializable
{
    private File rutaImagen;

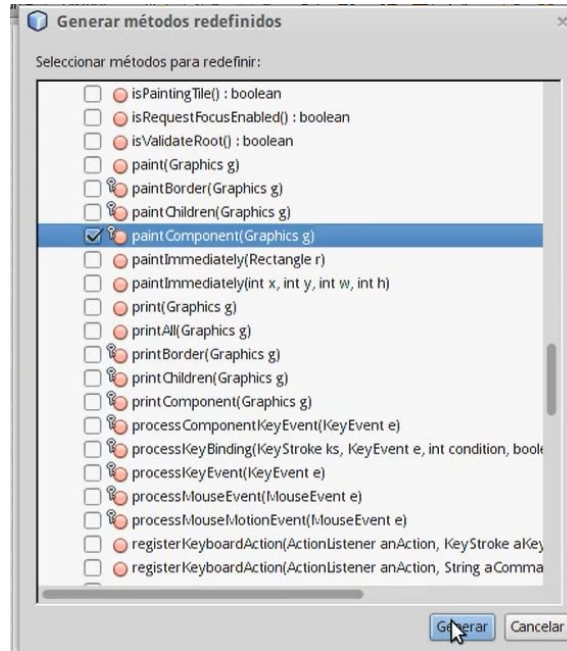
    public JPanelImagen()
    {
    }

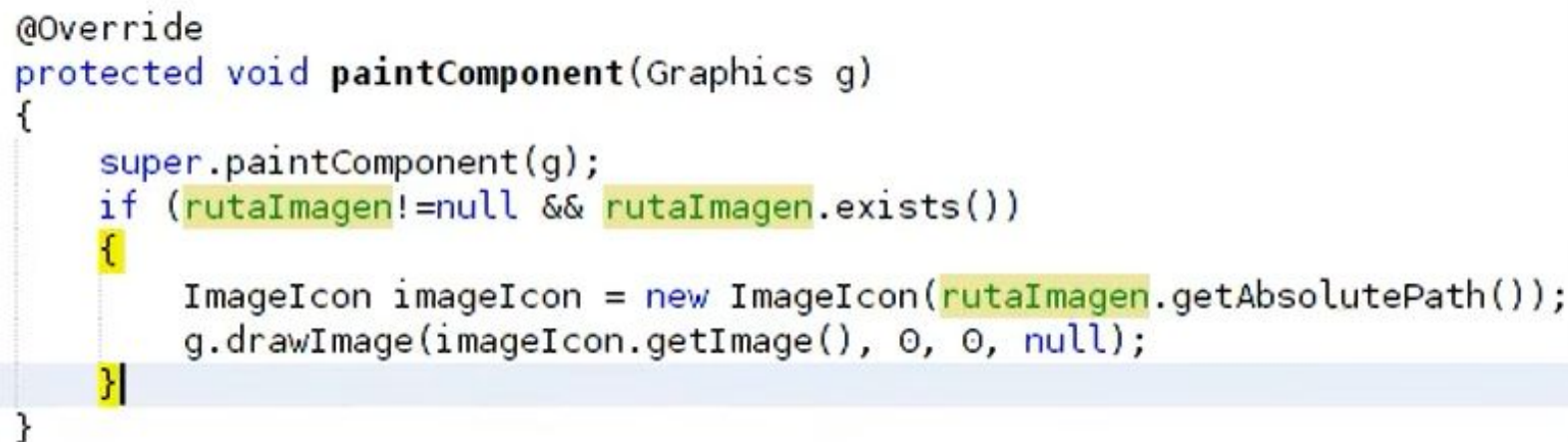
    public File getRutaImagen() {
        return rutaImagen;
    }

    public void setRutaImagen(File rutaImagen) {
        this.rutaImagen = rutaImagen;
    }
}
```

00:00:01

Añadimos el override del método paintComponent (Clase JComponent)

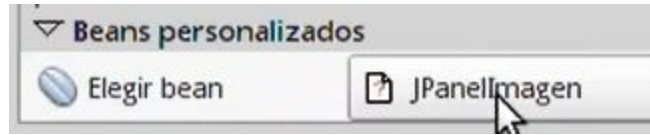


The image shows a snippet of Java code in an IDE. On the left, there is a vertical sidebar with a small square icon at the top and a bracket below it. The code is displayed in a light gray editor area. The code defines an overridden method `paintComponent` that calls the superclass method and then checks if `rutaImagen` is not null and exists. If it does, it creates an `ImageIcon` from the absolute path and draws it. The code is color-coded: keywords are blue, identifiers are black, and string literals are green. The closing brace of the `paintComponent` method is highlighted with a yellow background. A light blue horizontal bar is positioned below the closing brace of the `paintComponent` method.

```
@Override
protected void paintComponent(Graphics g)
{
    super.paintComponent(g);
    if (rutaImagen!=null && rutaImagen.exists())
    {
        ImageIcon imageIcon = new ImageIcon(rutaImagen.getAbsolutePath());
        g.drawImage(imageIcon.getImage(), 0, 0, null);
    }
}
```

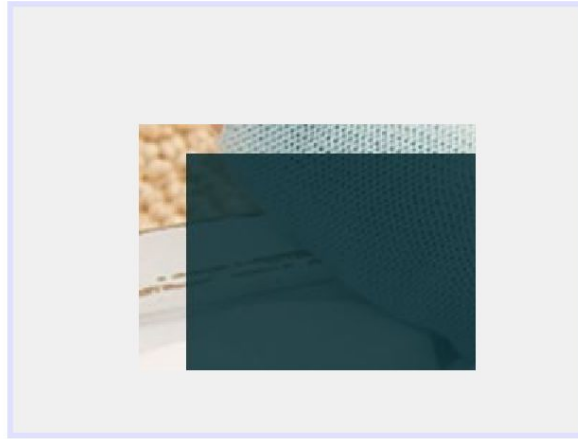
En la paleta, botón derecho gestionar paleta

Añadimos el .jar de nuestro proyecto en la paleta, en la carpeta beans personalizados.



Ya se puede probar

Incluimos el JPanel en un JFrame de prueba y añadimos un fichero en la propiedad.



Panel personalizado

En lugar de hacer una selección de fichero en la propiedad de netbeans, vamos a cambiar el tipo de atributo para lanzar nuestro panel personalizado.



Primero creamos una clase ImagenFondo

```
public class ImagenFondo implements Serializable{
```

```
    private File rutaImagen;  
    private Float opacidad;
```

```
    public ImagenFondo() {  
    }
```

```
    public File getRutaImagen() {  
        return rutaImagen;  
    }
```

```
    public void setRutaImagen(File rutaImagen) {  
        this.rutaImagen = rutaImagen;  
    }
```

```
    public Float getOpacidad() {  
        return opacidad;  
    }
```

```
    public void setOpacidad(Float opacidad) {  
        this.opacidad = opacidad;  
    }  
}
```

También constructor con 2 parámetros:

```
public ImagenFondo(File rutaImagen, Float opacidad) {  
    this.rutaImagen = rutaImagen;  
    this.opacidad = opacidad;  
}
```

Cambios en jPanelImagen

```
public class jPanelImagen extends JPanel implements Serializable {
    //Cambiamos
    // private File rutaImagen;
    private ImagenFondo imagenFondo;

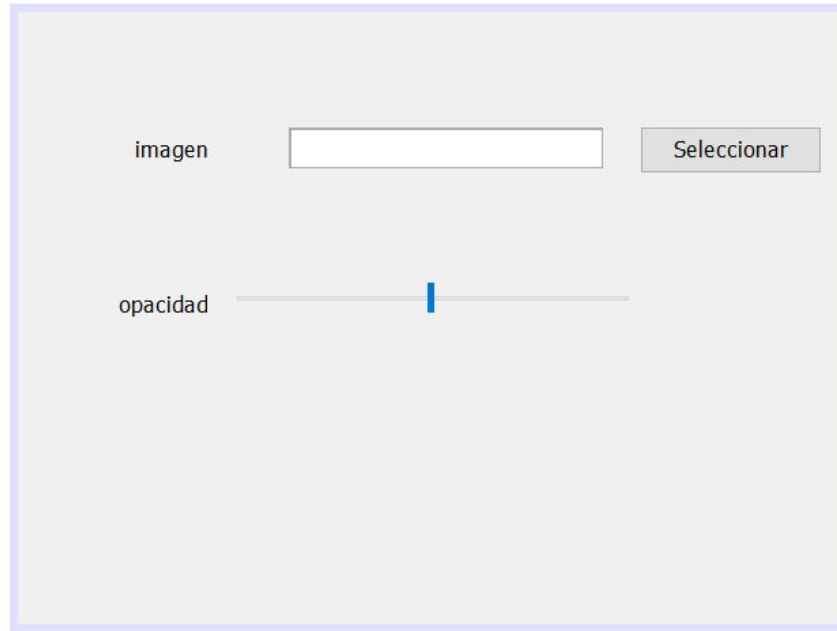
    public jPanelImagen() {
    }

    public ImagenFondo getImagenFondo() {
        return imagenFondo;
    }

    public void setImagenFondo(ImagenFondo imagenFondo) {
        this.imagenFondo = imagenFondo;
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g); //To change body of generated methods, choose Tools | Templates
        if(imagenFondo.getRutaImagen() != null && imagenFondo.getRutaImagen().exists()){
            ImageIcon imageIcon = new ImageIcon(imagenFondo.getRutaImagen().getAbsolutePath());
            g.drawImage(imageIcon.getImage(), 0, 0, null);
        }
    }
}
```

Creamos un JPanelForm llamado ImagenFondoPanel e insertamos jTextField, botón y slider



Botón seleccionar archivo

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    JFileChooser fileChooser = new JFileChooser();  
    int resultado = fileChooser.showOpenDialog(this);  
    if (resultado == JFileChooser.APPROVE_OPTION) {  
        File file = fileChooser.getSelectedFile();  
        jTextField1.setText(file.getAbsolutePath());  
    }  
}
```

Creamos método para acceder a los valores en JPanelImagen

```
public ImagenFondo getSelectedValue() {  
  
    ImagenFondo imagenFondo = new ImagenFondo(file, this.jSlider1.getValue());  
    return imagenFondo;  
}
```



Añadimos al pintar componente la opacidad

```
@Override
protected void paintComponent(Graphics g) {
    super.paintComponent(g); //To change body of generated methods, choose Tools | Templates.

    if (imagenFondo.getRutaImagen() != null && imagenFondo.getRutaImagen().exists())
    {
        ImageIcon imageIcon = new ImageIcon(imagenFondo.getRutaImagen().getAbsolutePath());

        Graphics2D g2d = (Graphics2D)g;
        g2d.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER, imagenFondo.getOpacidad()/100));

        g.drawImage(imageIcon.getImage(), 0, 0, null);
        g2d.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER, 1));|
    }
}
```


Creamos una nueva clase java para editar las propiedades

```
import java.beans.PropertyEditorSupport;
```

```
/**
 *
 * @author pablo
 */
public class ImagenFondoPropertyEditorSupport extends PropertyEditorSupport
{
}
```

Generar

Constructor...

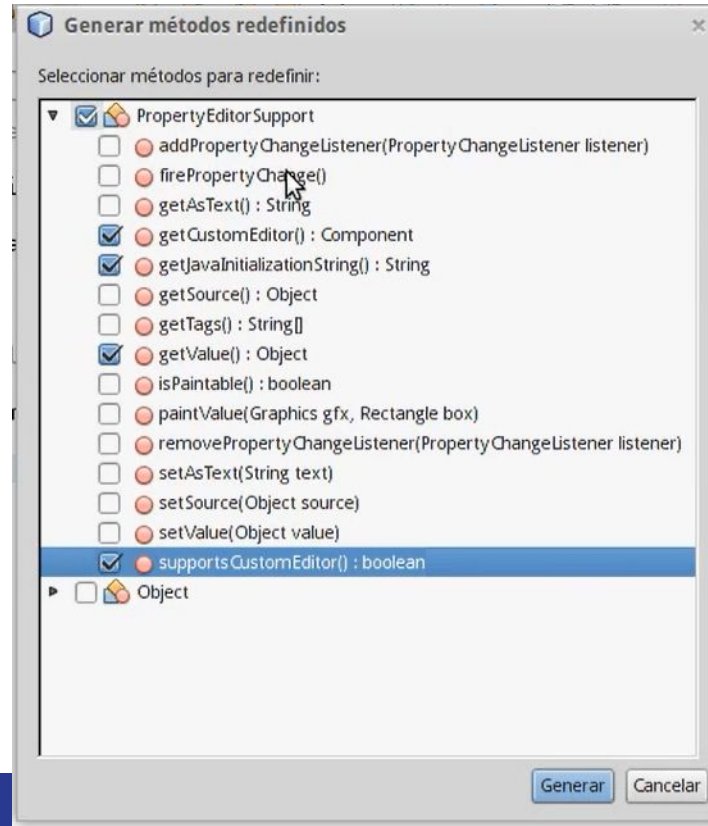
Logger...

toString()...

Redefinir método...

Agregar Propiedad...

Redefinimos métodos



```
public class ImagenFondoPropertyEditorSupport extends PropertyEditorSupport {
    private ImagenFondoPanel imagenFondoPanel= new ImagenFondoPanel();

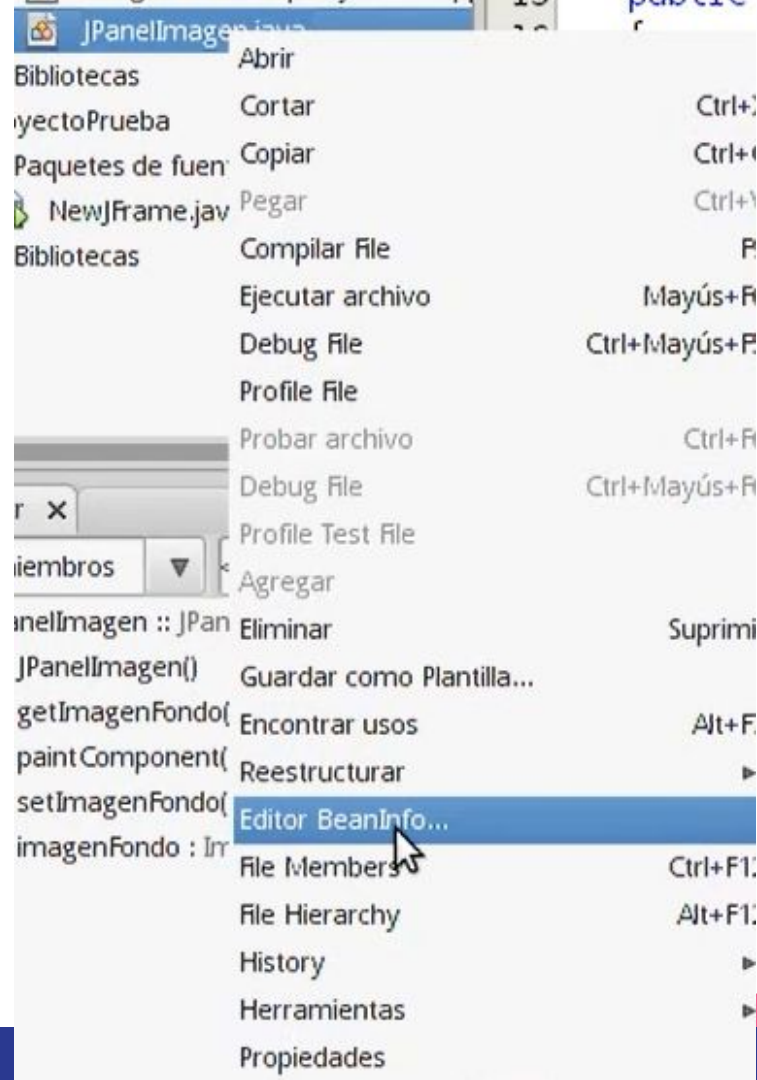
    @Override
    public boolean supportsCustomEditor() {
        //Si tiene un editor de propiedades personalizado
        return true; //To change body of generated methods, choose Tools | Templates.
    }

    @Override
    public Component getCustomEditor() {
        //Nos pide el panel
        return imagenFondoPanel; //To change body of generated methods, choose Tools | Templates.
    }

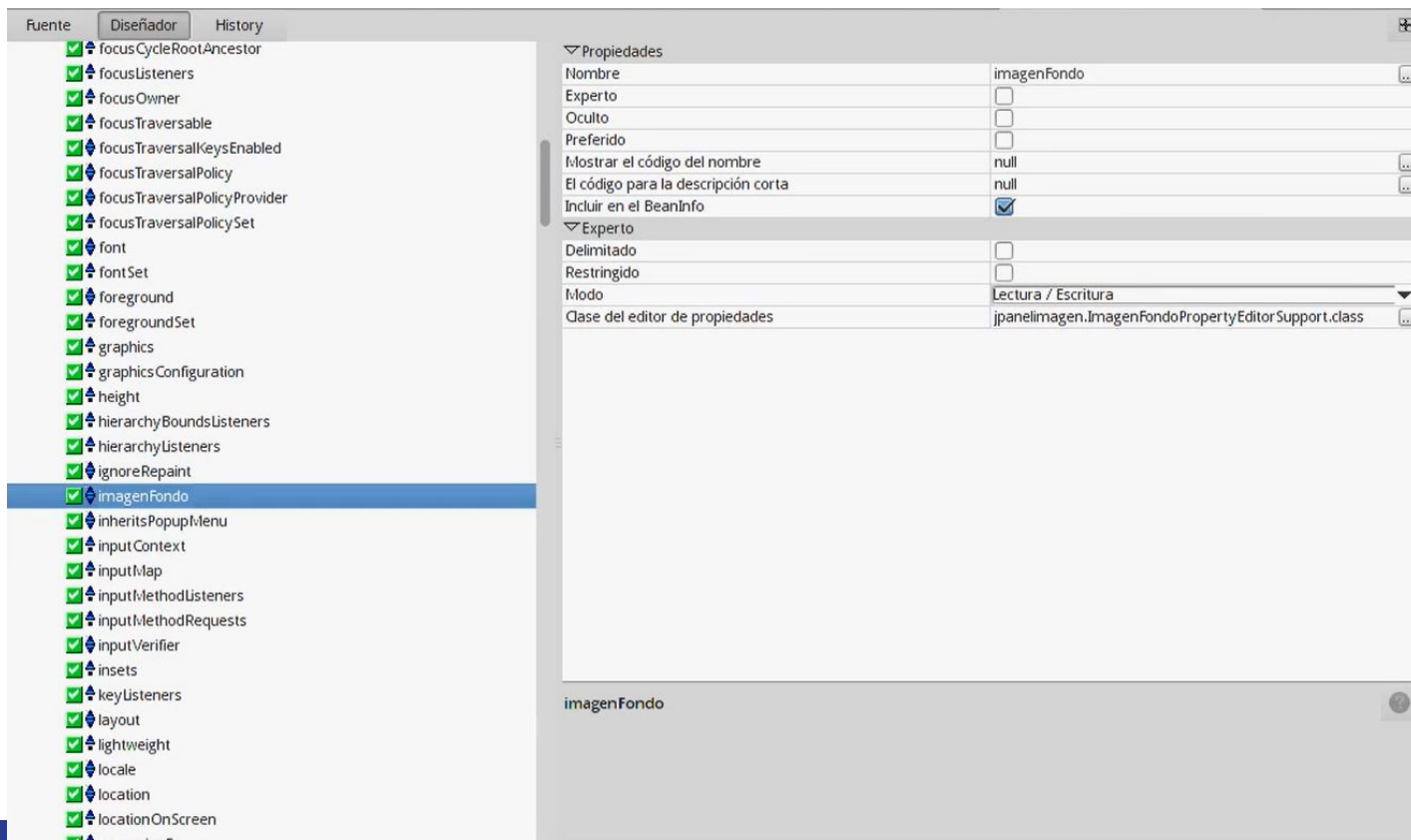
    @Override
    public String getJavaInitializationString() {
        return super.getJavaInitializationString(); //To change body of generated methods, choose T
    }

    @Override
    public Object getValue() {
        //Devuelve el valor (gracias al metodo que creamos para devolver nuestra imagenFondo
        return imagenFondoPanel.getSelectedValue(); //To change body of generated methods, choose T
    }
}
```

Editar BeanInfo



Cambiar clase del editor de propiedades



La opción Herramientas>Paleta>Componentes Swing/Awt permite modificar el contenido de la Paleta.

jPanelImagen1 [JPanelImagen] - imagenFondo

Establecer propiedad jPanelImagen1's imagenFondo utilizando: Editor predeterminado

Imagen:

...

Opacidad:

50

Aceptar

Cancelar

Ayuda

Lista

Barra de desplazamiento

Panel de desplazamiento

Panel

Lienzo

Barra de menú

Menú emergente

Beans personalizados

Elegir bean

JPanelImagen

Persistencia de Java

jPanelImagen1 [JPanelImagen] - Properties

Propiedades

Enlace

Eventos

Codigo

height	100	...
hierarchyBoundsListeners	<predeterminado>	...
hierarchyListeners	<predeterminado>	...
ignoreRepaint	<input type="checkbox"/>	...
imagenFondo		...
inheritsPopupMenu	<input type="checkbox"/>	...
inputContext	<predeterminado>	...
inputMap	<predeterminado>	...
inputMethodListeners	<predeterminado>	...
inputMethodRequests	<ninguna>	...

Completar por último para que al ejecutar el proyecto se complete bien la creación en el initComponents()

```
@Override
public String getJavaInitializationString() {
    ImagenFondo imagenFondo = this.imagenFondoPanel.getSelectedValue();
return "new jpanelimagen.ImagenFondo("+new java.io.File(\""+imagenFondo.getRutaImagen().getAbsolutePath().replace("\\", "\\/")+"\", "+imagenFondo.getOpacidad()+\"f\");
}
```