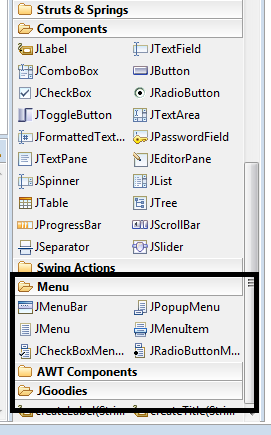
**MENÚS Y PESTAÑAS**

**Crear Menús**

Hay que crear un JMenuBar, la barra de menús.

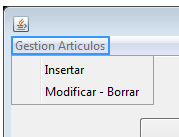
JMenu, van dentro del JMenuBar, son los menús de la barra de menús.

JMenuItem, los elementos de menú de un JMenu, van dentro de JMenu.

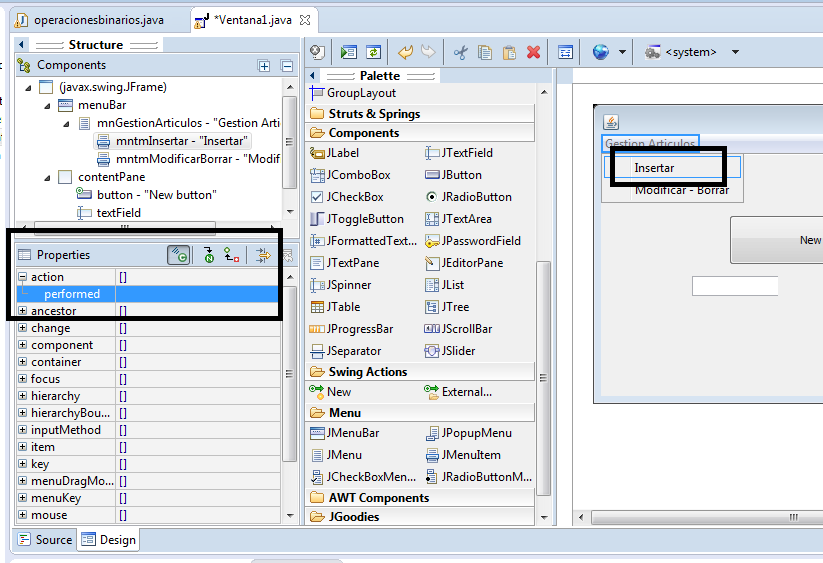


**Si queremos que el menú sea el mismo para todas las ventanas de la aplicación, se creará un JFrame con la barra de menú y sus menús. Y esa clase se heredará al resto, se pondrá como extend al resto de clases JFrame que se desea que hereden el menú. Por ejemplo Ventana2 hereda el JFrame con el menú llamado Mimenu**

**public** **class** Ventana2 **extends** Mimenu {….

****

Para añadir código a los menús, seleccionamos el elemento, y en las **Properties** seleccionamos los **eventos** y doble clic en ***Action -> performed***. Y dentro del código que se visualiza en el ***actionPerformed*** del ***ActionListener*** es donde se añadirá el código para realizar las acciones que se desean al pulsar el menú.

****

JMenuBar menuBar = **new** JMenuBar();

setJMenuBar(menuBar);

JMenu mnGestionArticulos = **new** JMenu("Gestion Articulos");

menuBar.add(mnGestionArticulos);

JMenuItem mntmInsertar = **new** JMenuItem("Insertar");

mntmInsertar.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

// Acciones del elemento de menú

}

});

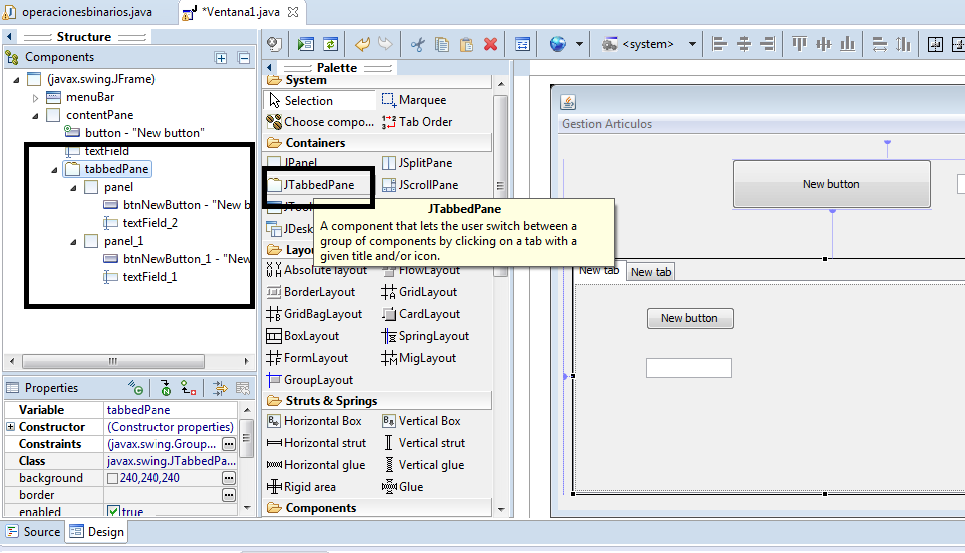
mnGestionArticulos.add(mntmInsertar);

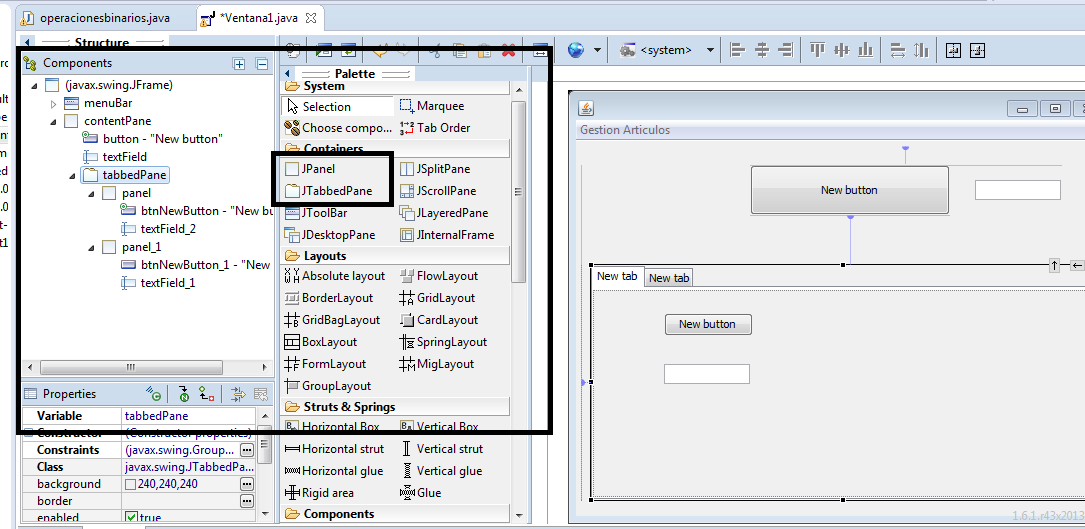
**Crear Tabs (Pestañas)**

Para crear pestañas.

Primero se añade un ***JTabbedPane*** y dentro de este se añaden ***JPanel***, tantos como pestañas.

A los ***JPanel*** hay que marcarles con ***Absolute Layout*** para poder colocar los elementos en cualquier posición del **JPanel**.

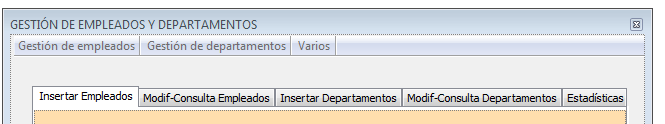
****

****

**Para detectar que se selecciona una pestaña creamos un evento Change**

****

**Para trabajar con la pestaña:**



**//Seleccionar y mostrar una pestaña concreta (empieza la numeración en 0, esto selecciona la 2):**

tabbedPane.setSelectedIndex(2);

tabbedPane.setVisible(**true**);

**// DETECCIÓN DE PULSAR EN UNA PESTAÑA U OTRA**

tabbedPane.addChangeListener(new ChangeListener() {

   public void stateChanged(ChangeEvent e) {

      if (tabbedPane.getSelectedIndex() == 0) {

// INSERCION EMPLES acciones

     } else if (tabbedPane.getSelectedIndex() == 1) {

// MODIF EMPLES acciones

     } else if (tabbedPane.getSelectedIndex() == 2) {

// INSERCION DEP acciones

     } else if (tabbedPane.getSelectedIndex() == 3) {

// MODIF DEP acciones

     } else

          // ESTADISTICAS

           PintarCalculos();

  }

});//fin detección pulsar en una pestaña u otra

**Abrir sistema de archivos para seleccionar archivo. Crear JFileChooser**

http://chuwiki.chuidiang.org/index.php?title=JFileChooser

**JFileChooser** es una clase java que nos permite mostrar fácilmente una ventana para la selección de un fichero. Si queremos abrirlo para leer el fichero, podemos llamarlo así

JFileChooser fileChooser = new JFileChooser();

int seleccion = fileChooser.showOpenDialog(ventana);

ventana es cualquier componente java de alguna de nuestras ventanas (JFrame, JDialog ..), la que queramos que haga de ventana padre del diálogo JFileChooser. Nuestro diálogo JFileChooser no se podrá ir detrás de su ventana padre.

Ejemplo de un JFileChooser



Código de prueba:

**private** **void** probar() {

String path = "";

JFileChooser fileChooser = **new** JFileChooser();

//Ponemos título al diálogo que se visualiza

fileChooser.setDialogTitle("Selecciona el fichero.");

// Seleccionamos el directorio para iniciar la búsqueda y que lo abra

File miDir = **new** File("./imagenes");

// Asignamos ese directorio, para que empiece a buscar ahí los ficheros.

fileChooser.setCurrentDirectory(miDir);

// Visualizamos el diálogo de elegir fichero

fileChooser.setVisible(**true**);

// Se espera que se seleccione fichero o se pulse cancelar

**int** seleccion = fileChooser.showOpenDialog(**this**);

**if** (seleccion == JFileChooser.***APPROVE\_OPTION***) {

// Pulsa aceptar

File fichero = **null**;

//Se selecciona un fichero y se devuelve a un File

fichero = fileChooser.getSelectedFile();

System.***out***.println("archivo:" + fichero.toString());

//Lo pasamos a string si queremos el path

path = fichero.toString();

System.***out***.println("Fichero seleccionado " + path);

}

**if** (seleccion == JFileChooser.***CANCEL\_OPTION***) {

// Pulsa cancelar o salida

System.***out***.println("Ha pulsado Cancelar ");

}

}

La llamada **showOpenDialog**() se quedará bloqueada hasta que el usuario elija un fichero y cierre la ventana. A la vuelta, en selección tendremos

* **JFileChooser.CANCEL\_OPTION** Si el usuario le ha dado al botón cancelar.
* **JFileChooser.APPROVE\_OPTION** Si el usuario le ha dado al botón aceptar
* **JFileCHooser.ERROR\_OPTION** Si ha ocurrido algún error.

Comprobando que se ha dado al botón aceptar, podemos obtener el fichero seleccionado por el usuario así

if (seleccion == JFileChooser.APROVE\_OPTION)

{

File fichero = fileChooser.getSelectedFile();

// Aquí debemos abrir y leer el fichero.

...

}

Para seleccionar un fichero para guardar datos, el mecanismo es igual, pero se llama al método **showSaveDialog()**

JFileChooser fileChooser = new JFileChooser();

int seleccion = fileChooser.showSaveDialog(areaTexto);

if (seleccion == JFileChooser.APPROVE\_OPTION)

{

File fichero = fileChooser.getSelectedFile();

// Aquí debemos abrir el fichero para escritura

// y salvar nuestros datos.

...

}

La única diferencia entre uno y otro es la etiqueta del diálogo y de los botones. Uno pondrá "Abrir" y otro "Guardar"

**Trabajar con Imágenes:**

Una imagen la cargamos en un JLabel de la siguiente manera:

JLabel foto = **new** JLabel("FOTO");

// path de la Imagen a Cargar

String path = "./imagenes/estuchecompleto.jpg";

//Creo una ImageIcon con el path de la imagen

ImageIcon fot = **new** ImageIcon(path);

//Creo un Icon con los datos del ImageIcon, y con el ancho y el alto que haya asignado a mi Jlabel foto, y se indica que se ajuste al tamaño de la label con SCALE\_SMOOTH

Icon icon = **new** ImageIcon(

fot.getImage().getScaledInstance(foto.getWidth(),

foto.getHeight(), Image.***SCALE\_SMOOTH***));

System.***out***.println("Imagen a cargar " + path);

//Asignamos el icono a la label foto, y se dibuja de nuevo

foto.setIcon(icon);

foto.repaint();

**Grabar una imagen en MySQL.**

**Una tabla con un campo autoincrement y una imagen**

CREATE TABLE `ejemplo`.`pruebaimagen` ( `cod` INT NOT NULL primary key AUTO\_INCREMENT, `imagen` LONGBLOB NOT NULL ) ENGINE = InnoDB

[CREATE](http://localhost/phpmyadmin/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/create-table.html) [TABLE](http://localhost/phpmyadmin/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/create-table.html) `ejemplo`.`pruebaimagen` ( `cod` [INT](http://localhost/phpmyadmin/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/numeric-types.html) [NOT](http://localhost/phpmyadmin/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/logical-operators.html#operator_not) NULL primary key AUTO\_INCREMENT, `imagen` [LONGBLOB](http://localhost/phpmyadmin/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/string-types.html) [NOT](http://localhost/phpmyadmin/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/logical-operators.html#operator_not) NULL ) ENGINE = InnoDB

El tipo de dato debe ser un LONGBLOB, que admite hasta 4GB.

También depende de los parámetros de configuración de mysql (my.ini) **max\_allowed\_packet** e **innodb\_log\_file\_size**

Para Almacenar imágenes muy grandes, estos valores del fichero de configuración de mysql (my.ini) max\_allowed\_packet, innodb\_log\_file\_size los ponemos a 512M por ej.

El código para grabar es el siguiente:

String insert = "insert into pruebaimagen (imagen) values(?)";

FileInputStream ficheroimagen = **null**;

PreparedStatement sentencia = **null**;

**try** {

Class.*forName*("com.mysql.jdbc.Driver");

Connection conexion =

DriverManager.*getConnection*("jdbc:mysql://localhost/ejemplo", "root", "");

File file = **new** File(path);

// Cargo el fichero en un FileInputStream

ficheroimagen = **new** FileInputStream(file);

sentencia = conexion.prepareStatement(insert);

// Guardo la imagen, utilizo el método setBinaryStream, al final

// indico la longitud en bytes del fichero

sentencia.setBinaryStream(1, ficheroimagen, (**int**) file.length());

sentencia.executeUpdate();

sentencia.close();

conexion.close();

System.***out***.println("\*\*REGISTRO GRABADO\*\*");

} **catch** (ClassNotFoundException cn) {

System.***out***.println("Error DRIVER." + cn.getMessage());

} **catch** (SQLException e) {

System.***out***.println("Error SQL." + e.getMessage());

} **catch** (FileNotFoundException e) {

System.***out***.println("Error no encuentro la imagen." + e.getMessage());

}

**Leer la imagen de MySQL y cargarla en una JLabel**

**try** {

Class.*forName*("com.mysql.jdbc.Driver");

Connection conexion =

DriverManager.*getConnection*("jdbc:mysql://localhost/ejemplo", "root", "");

Statement sentencia = conexion.createStatement();

ResultSet resul = sentencia.executeQuery

("Select \* from pruebaimagen where codarti = 1");

//Sólo debe haber 1

**if** (resul.next()) {

**int** codi = resul.getInt(1);

//La imagen se carga en un tipo dato Blob

Blob blob = resul.getBlob(4);

**float** pp = resul.getFloat(3);

String nom = resul.getString(2);

System.***out***.println(" Cod=" + codi + ", denom= " + nom);

//Preparo la carga de la imagen,

//utilizao un array de bytes y la imagen se carga ahí

**byte**[] data = **null**;

//Cargo en data los bytes de la posición 1 hasta la longitud

data = blob.getBytes(1, (**int**) blob.length());

Icon icon=**null**;

**try** {

//Convierte la imagen a un BufferedImage

BufferedImage img = ImageIO.read(**new** ByteArrayInputStream(data));

//Convierto la imagen BufferedImage a un ImageIcon

ImageIcon imgIcon = **new** ImageIcon(img);

icon = **new**

ImageIcon(imgIcon.getImage().getScaledInstance(foto.getWidth(),

foto.getHeight(), Image.***SCALE\_SMOOTH***));

System.***out***.println(" icono= " + icon.toString());

System.***out***.println(" Icon altura:" + icon.getIconHeight());

foto.setIcon(icon);

foto.repaint();

} **catch** (IOException ex) {

System.***out***.println("Error al cargar la imagen." + ex.getMessage());

}

}

} **catch** (ClassNotFoundException cn) {

System.***out***.println("Error SQL." + cn.getMessage());

} **catch** (SQLException e) {

System.***out***.println("Error SQL." + e.getMessage());

}