

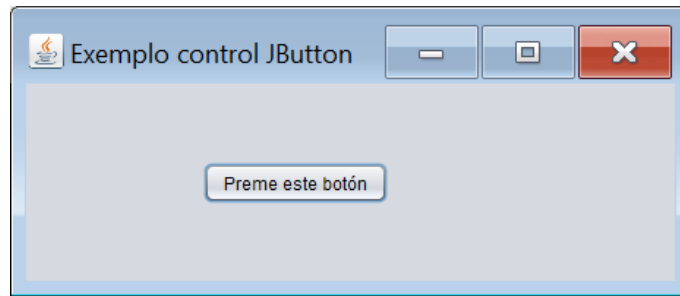
1. Xestión de compoñentes básicos

Cando desenvolvemos aplicacións gráficas de usuario cada un dos formularios que compoñen as nosas aplicacións están compostos por unha serie de compoñentes gráficos que son empregados para levar a cabo a comunicación de información entre o usuario e a aplicación. Existe unha grande variedade de compoñentes gráficos, sendo a súa complexidade e a súa utilidade moi variable en función do compoñente. Non obstante, na maior parte das aplicacións empréganse unha serie de compoñentes gráficos básicos que son sempre os mesmos. Empréganse para a comunicación de informacións sinxelas entre o usuario e a aplicación. Pasaremos agora a estudar estes compoñentes gráficos básicos. Cabe dicir que centrarémonos nas tarefas para as que se soen empregar estes compoñentes máis habitualmente, xa que estudar calquera destes compoñentes en profundidade sería un traballo por un lado excesivo debido a que son moitísimas as propiedades, eventos e métodos que xestiona cada un deles, e por outra banda sería unha perda de tempo, xa que estudaríamos funcionalidades que rara vez vanse empregar. É mellor idea saber xestionar cada compoñente para empregar as súas funcionalidades habituais e no caso de ter que empregar algunha funcionalidade excepcional, aprender como xestionar esa funcionalidade excepcional en concreto. Para coñecer tódalas propiedades, eventos e métodos de cada un dos compoñentes básicos sempre poderemos acceder á API (application programming interface) de cada compoñente. De seguido imos enumerar os diferentes compoñentes básicos que imos estudar:

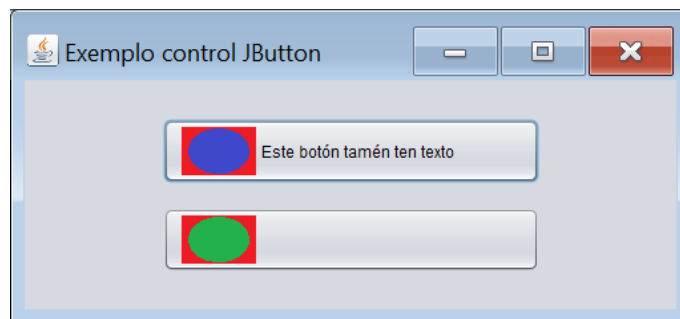
- Botón
- Etiqueta
- Caixa de texto
- Casilla de verificación
- Botón de opción
- Área de texto

2. Compoñente Botón (JButton)

Un botón é un compoñente ao cal pódesele asociar unha acción que será realizada no caso de que fagamos clic sobre el. Cando precisamos de colocar un botón nas nosas aplicacións, o compoñente que debemos de empregar é o compoñente JButton. O compoñente botón defínese a través da clase `javax.swing.JButton`. Graficamente o seu aspecto é o seguinte:



Como pódese observar na imaxe, o botón ten asociado un texto (Preme este botón). Ao pulsar sobre o botón poderemos desencadear unha acción que teremos programado previamente. A través das propiedades do JButton é posible persoalizar o seu aspecto de xeito que o adaptemos aos requirimentos das nosas aplicacións:



Propiedades do control JButton empregadas máis habitualmente e que pódense establecer a través do entorno de programación:

Propiedade	Función
Background	Cor de fondo do compoñente
Cursor	Mediante esta propiedade indícase a imaxe que amosará o punteiro do rato ao navegar sobre o compoñente
Enabled	Se esta propiedade está activada o compoñente será funcional. No caso de que esta propiedade estea desactivada o compoñente será visualizable, pero o usuario non poderá interaccionar con el.
Focusable	Se esta propiedade está activada o compoñente entrará na roda de reparto do foco de xeito que ao premir a tecla de cambio de foco (habitualmente o tabulador) nalgún momento tomará o foco da aplicación (cando sexa a súa quenda na roda de reparto do foco). Pola contra, se esta propiedade está desactivada o compoñente non entrará na roda de reparto do foco e soamente será posible acceder a el mediante o rato.
Font	Características da fonte do compoñente (tipo de fonte, tamaño, ...)
Foreground	Cor do texto do compoñente
HorizontalTextPosition	Localización horizontal do texto do compoñente respecto á imaxe amosada polo compoñente
Icon	Mediante esta propiedade indícase a imaxe a amosar no compoñente. Se non indicamos nada o compoñente amosará o seu aspecto estándar.
IconTextGap	Separación entre o texto do compoñente e a imaxe empregada polo compoñente
PressedIcon	Mediante esta propiedade indícase a imaxe a amosar cando prememos sobre o compoñente.
RollOverEnabled	Se esta propiedade está desactivada, cando pasemos sobre o compoñente amosarase a imaxe indicada na súa propiedade RollOverIcon. Se RollOverIcon non está definida non se amosará ningunha imaxe. Pola contra, se a propiedade RollOverEnabled está activada, cando pasemos sobre o compoñente amosarase a imaxe indicada na súa propiedade RollOverIcon. Se RollOverIcon non está definida amosarase a imaxe definida na súa propiedade SelectedIcon e se a propiedade SelectedIcon non está definida non se amosará nada.

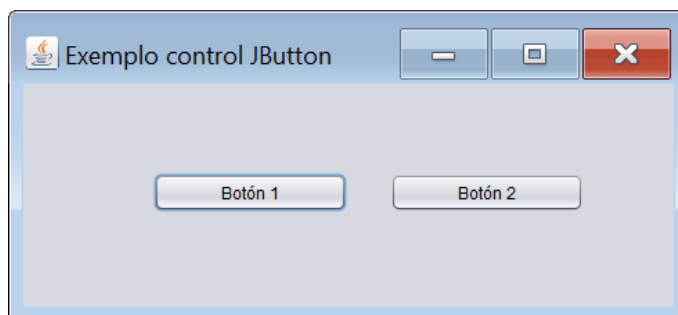
RollOverIcon	Mediante esta propiedad indícase a imaxe a amosar cando pasamos co rato sobre o compoñente.
SelectedIcon	Mediante esta propiedad indícase a imaxe a amosar cando o compoñente está seleccionado.
Text	Empregase para establecer o texto que aparece no botón
ToolTipText	Mediante esta propiedad establécese unha mensaxe (tooltip) que será amosado ao deixar o punteiro do rato sobre o compoñente. É habitual empregar esta mensaxe para indicar cal é a utilidade do compoñente
VerticalTextPosition	Localización vertical do texto do compoñente respecto á imaxe amosada polo compoñente

Eventos do control JButton empregados máis habitualmente e que pódense establecer a través do entorno de programación:

Evento	Lanzamento
ActionPerformed	Este evento é disparado no momento no que facemos clic sobre o botón

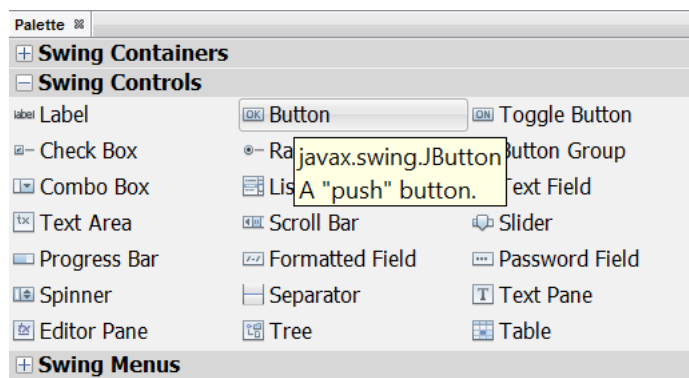
Xestión de botóns empregando NetBeans

A continuación imos desenvolver o seguinte formulario:

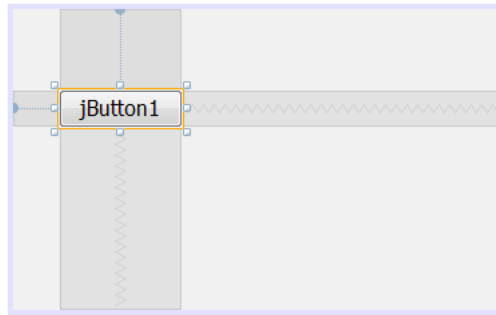


Ao facer clic sobre o botón 1 a aplicación deberá mostrar unha mensaxe indicando que se premeu o botón 1. Ao facer clic sobre o botón 2 deberá mostrar unha mensaxe indicando que se pulsou o botón 2.

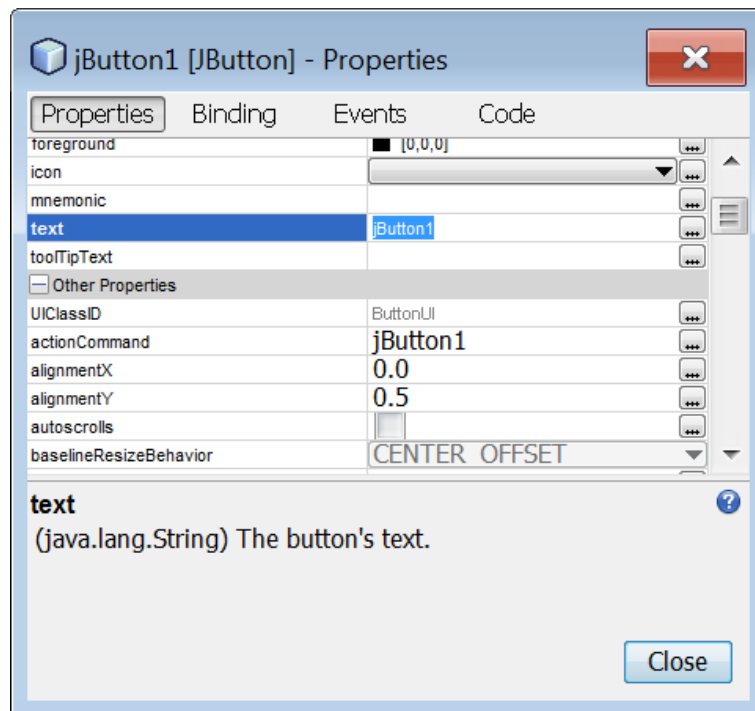
O primeiro que temos que facer é engadir un compoñente de tipo JButton no noso formulario. Para elo primeiramente o seleccionamos na paleta de compoñentes do entorno de programación:



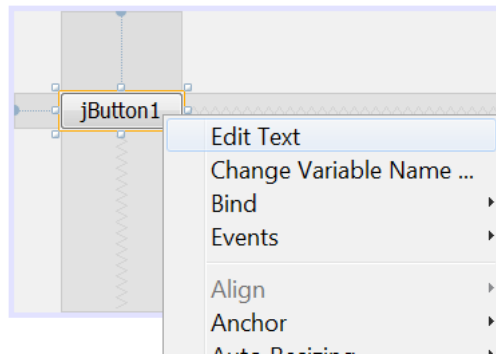
e o arrastramos ata o formulario:



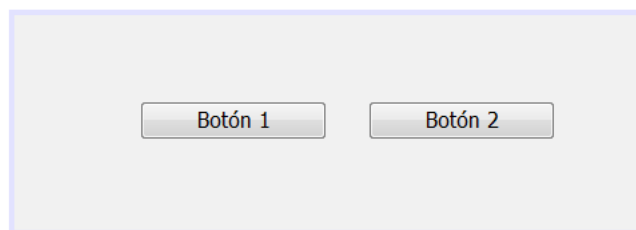
Unha vez situado dentro do formulario o adaptamos para que teña o aspecto visual que desexamos que teña. Para elo modificaremos o seu tamaño e o colocaremos na posición na que desexamos que estea situado. Por último modificaremos o texto que queremos que apareza no botón. Para modificar o texto do botón deberemos modificar a súa propiedade text. Podemos facelo de dous xeitos. A primeira é a través da xanela de propiedades do botón creado, accedendo á súa propiedade text e modificándoa:



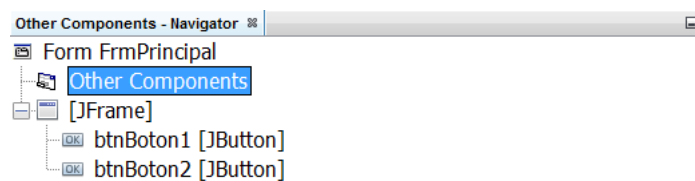
O outro xeito é facer clic co botón dereito do rato sobre o botón do cal queremos modificar a súa propiedade text. Alo facelo despregarase un menú contextual no cal deberemos de seleccionar a opción Edit Text. Ao facelo poderemos trocar o texto directamente sobre o compoñente:



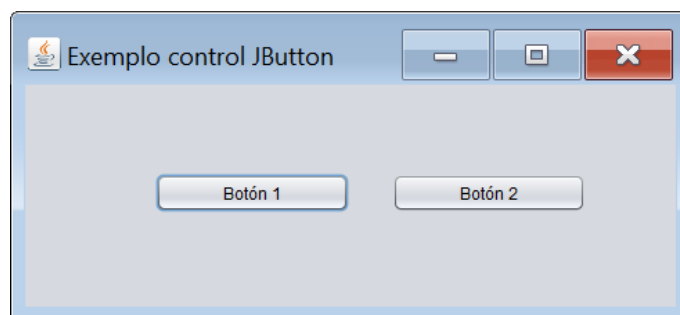
Unha vez configurado o primeiro botón, engadiremos un segundo botón ao formulario e tamén o configuraremos. Este será o resultado do deseño:



O noso formulario quedará configurado do seguinte xeito:



Se executamos o noso formulario este será o seu aspecto:



Como pódese observar, visualmente é o obxectivo buscado. Non obstante, non ten engadida ningunha funcionalidade. O formulario contén dous botóns que non facen absolutamente nada. A continuación imos ver como engadir funcionalidade aos botóns.

Para detectar un evento de tipo clic sobre un botón hai que engadirlle ao botón unha interface de tipo `ActionListener` e implementar o seu método `actionPerformed`, indicando neste último que é o que queremos facer cando

fagamos clic sobre o botón. A pesar de ser unha tarefa mecánica é un traballo tedioso e é aquí onde o entorno de programación ven a facilitarnos as cousas. O único que teremos que facer para indicar que código será executado ao facer clic sobre un botón é facer dobre clic sobre el. Por exemplo, imos ver o proceso a seguir para indicar que ocorrerá cando fagamos clic sobre o JButton btnBoton1:

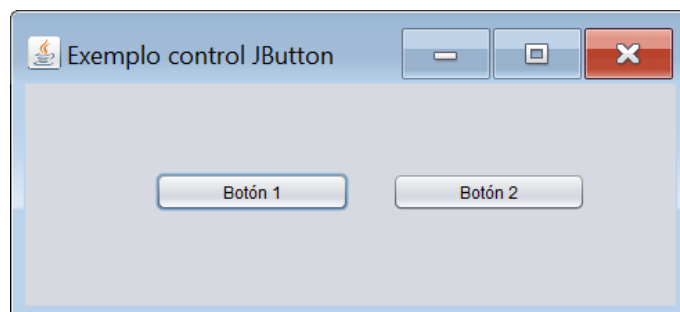
Primeiro facemos dobre clic sobre o btnBoton1 (pódese facer sobre a xanela de deseño do formulario o sobre o navegador de obxectos). Ao realizar esta acción amósasenos o código fonte do formulario que estamos creando e o cursor sitúase dentro dun método chamado btnBoton1ActionPerformed:

```
private void btnBoton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
}
```

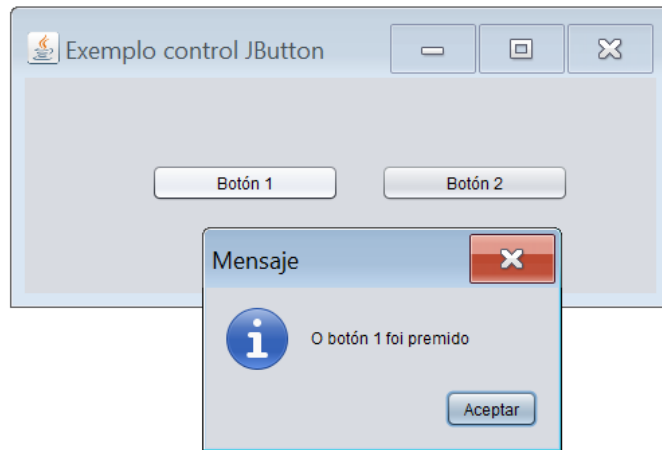
Como pódese observar o nome do método non é casual. A primeira parte do nome do método indica o control que xera o evento, mentres que a segunda parte do nome do método indica cal é o evento xestionado. O entorno de programación encárgase de engadir e implementar a interface e o método necesario para que ao facer clic sobre o botón btnBoton1 o método btnBoton1ActionPerformed sexa executado. Como podemos ver a parte tediosa da xestión de eventos é levada a cabo polo entorno de programación automaticamente. Agora unicamente deberemos de escribir neste método que é o que queremos que se faga cando fagamos clic co rato sobre o botón btnBoton1. Neste caso queremos que se amose unha mensaxe por pantalla indicando que o botón 1 foi premido:

```
private void btnBoton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    JOptionPane.showMessageDialog(this, "O botón 1 foi premido");  
}
```

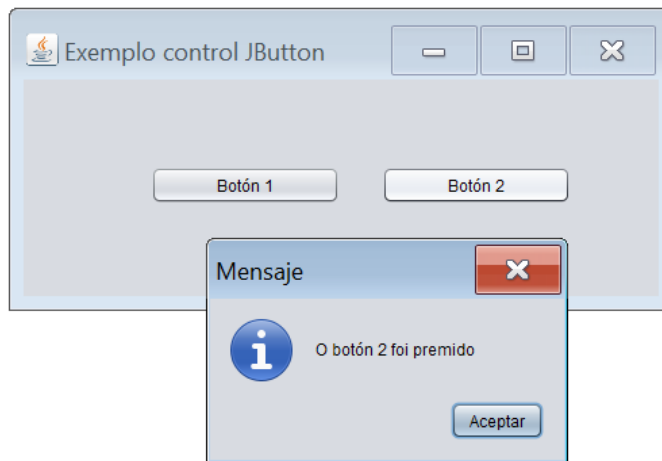
Unha vez implementado o código fonte das accións a realizar cando facemos clic sobre o botón 1, pasaremos a repetir o proceso sobre o botón 2. O resultado será a aplicación buscada tanto desde un punto de vista gráfico como desde un punto de vista funcional:



Ao pulsar sobre o botón 1:

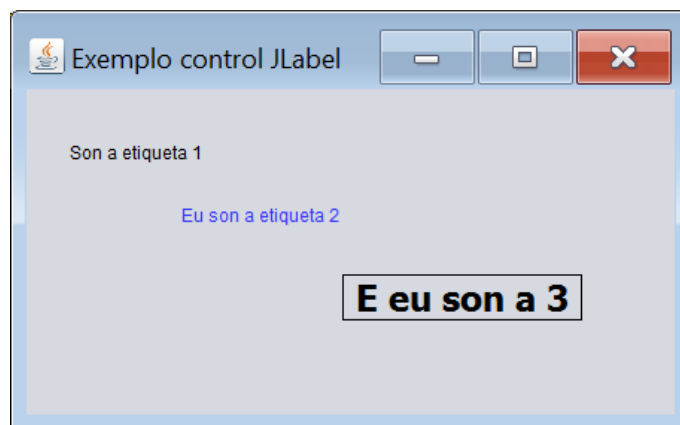


Ao pulsar sobre el botón 2:



3. Compoñente Etiqueta (JLabel)

Cando precisamos de colocar un texto estático dentro dos nosos formularios o compoñente que debemos empregar é o compoñente JLabel. Non é habitual que as etiquetas desencadeen eventos en función de accións realizadas sobre elas (aínda que é posible facelo, p.e. ao facer dobre clic sobre unha etiqueta). A súa función principal é amosar información de tipo estático nos nosos formularios. Normalmente acompaña a outros compoñentes e simplemente da información ao usuario de para que empregase ese compoñente acompañado. Outra utilidade que acostuma a dárselle ás etiquetas é empregalas para amosar imaxes dentro dos nosos formularios. O compoñente etiqueta defínese a través da clase `javax.swing.JLabel`. Graficamente o seu aspecto é o seguinte:



Como pódese observar na imaxe, o formulario contén 3 etiquetas, cada unha das cales amosa un texto. Indicamos anteriormente que as etiquetas se empregan como un elemento informativo, pero como pódese observar no exemplo tamén teñen un forte compoñente de persoalización que nos permite adaptalas ás necesidades gráficas dos nosos interfaces de usuario.

Propiedades do control JLabel empregadas máis habitualmente e que pódense establecer a través do entorno de programación:

Propiedade	Función
Border	Borde do compoñente
Cursor	Mediante esta propiedade indícase a imaxe que amosará o punteiro do rato ao navegar sobre o compoñente
Enabled	Se esta propiedade está activada o compoñente será funcional. No caso de que esta propiedade estea desactivada o compoñente será visualizable, pero o usuario non poderá interaccionar con el.
Focusable	Se esta propiedade está activada o compoñente entrará na roda de reparto do foco de xeito que ao premir a tecla de cambio de foco (habitualmente o tabulador) nalgún momento tomará o foco da aplicación (cando sexa a súa quenda na roda de reparto do foco). Pola contra, se esta propiedade está desactivada o compoñente non entrará na roda de reparto do foco e soamente será posible acceder a el mediante o rato.
Font	Características da fonte do compoñente (tipo de fonte, tamaño, ...)
Foreground	Cor do texto do compoñente
HorizontalAlignment	Aliñación horizontal do texto do compoñente
HorizontalTextPosition	Localización horizontal do texto do compoñente respecto á imaxe amosada polo compoñente
Icon	Mediante esta propiedade indícase a imaxe a amosar no compoñente. Se non indicamos nada o compoñente amosará o seu aspecto estándar.
IconTextGap	Separación entre o texto do compoñente e a imaxe empregada polo compoñente
Text	Empregase para establecer o texto que aparece na etiqueta
ToolTipText	Mediante esta propiedade establécese unha mensaxe (tooltip) que será amosado ao deixar o punteiro do rato sobre o compoñente. É habitual empregar esta mensaxe para indicar cal é a utilidade do compoñente
VerticalAlignment	Aliñación vertical do texto do compoñente
VerticalTextPosition	Localización vertical do texto do compoñente respecto á imaxe amosada polo compoñente

Eventos do control JLabel empregados máis habitualmente e que pódense establecer a través do entorno de programación: como xa dixéramos non é habitual xerar eventos a través das accións realizadas sobre as etiquetas, aínda

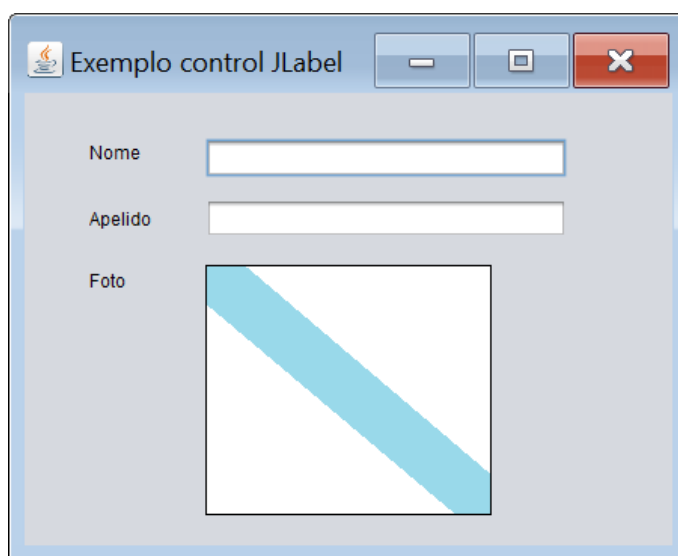
que nada impídenos escribir un método asociado a un evento xerado por unha etiqueta, p.e.: ao facer dobre clic sobre a etiqueta.

Métodos do control JLabel empregados máis habitualmente:

Método	Función
<code>public String getText()</code>	Emprégase para recuperar o texto dunha etiqueta
<code>public void setText(String text)</code>	Emprégase para establecer o texto dunha etiqueta

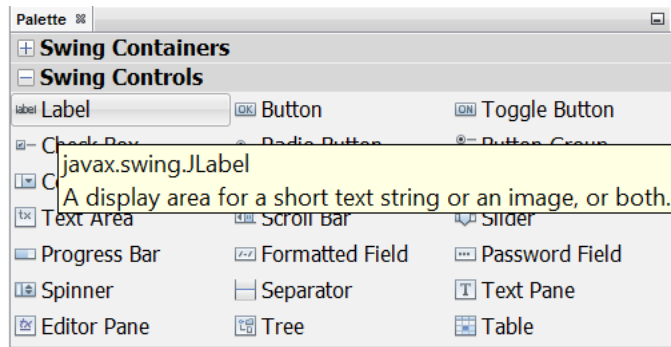
Xestión de etiquetas empregando NetBeans

A continuación imos desenvolver o seguinte formulario:

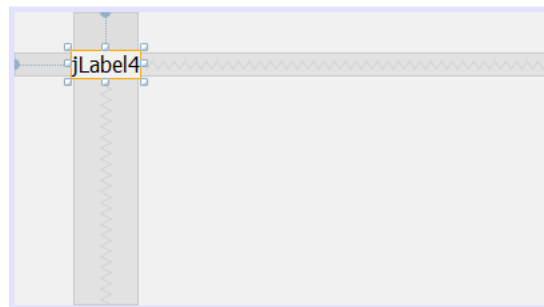


O formulario consta de catro etiquetas: nome, apelido, foto e unha etiqueta que imos empregar para amosar unha imaxe. As dúas primeiras etiquetas (nome e apelido) empréganse para acompañar a dous caixas de texto (o compoñente caixa de texto verémolo máis adiante. Indicar unicamente que adoita empregase para pedir información ao usuario), e indican ao usuario que tipo de información espérase que introduza nesas caixas de texto. A terceira etiqueta (foto) acompaña á etiqueta que contén unha imaxe e simplemente da unha descrición acerca do compoñente que acompaña. A carta etiqueta empregámola para amosar o funcionamento das etiquetas como contedores de imaxes.

Para engadir un compoñente de tipo JLabel no noso formulario primeiramente seleccionáremolo da paleta de compoñentes do entorno de programación:



e arrastrámolo ata o formulario:



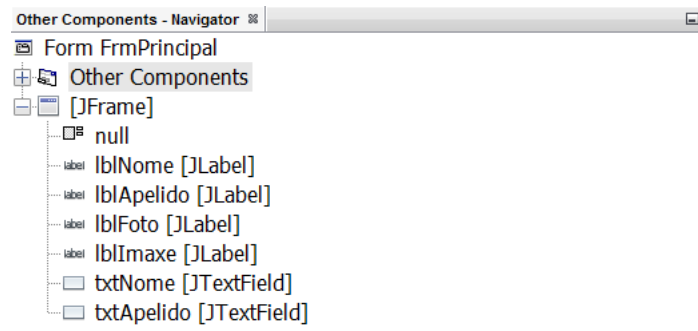
Unha vez situado dentro do formulario o adaptamos para que teña o aspecto visual que desexamos que teña. Para elo modificaremos o seu tamaño o colocaremos na posición na que desexamos que esté situado e modificaremos o texto que queremos que apareza no compoñente. Por ultimo poderemos modificar as súas propiedades para adaptar o seu aspecto ao aspecto que desexamos que teña o control na nosa aplicación.

Seguindo o noso exemplo, imos colocar catro etiquetas no noso formulario e dous JTextField (caixa de texto).

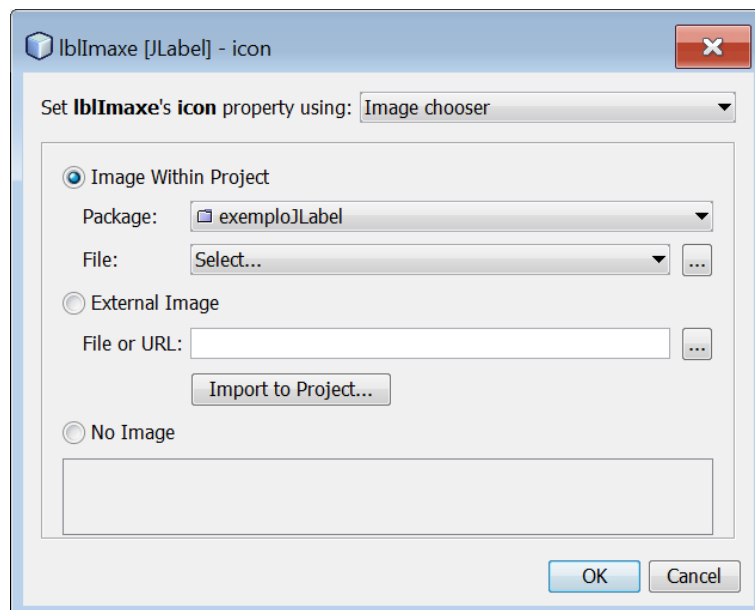
Unha vez engadidos e configurados os compoñentes, este será o resultado do deseño:



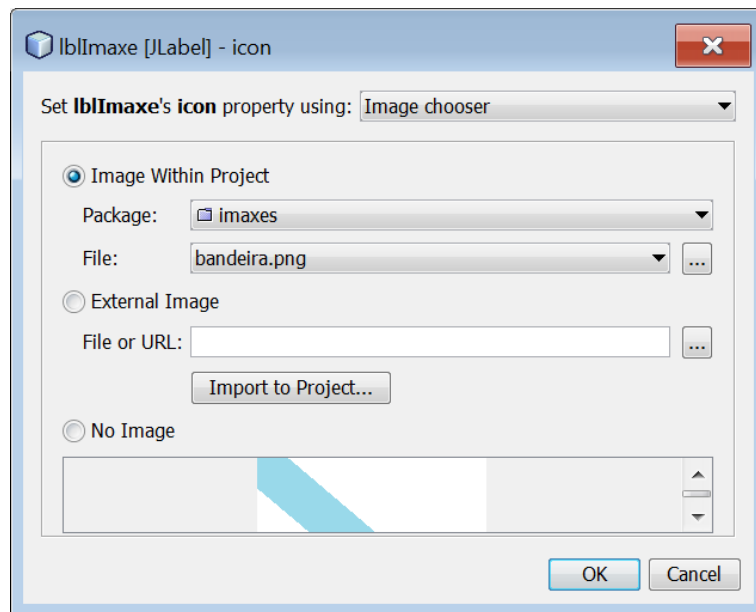
O noso formulario quedará configurado do seguinte xeito:



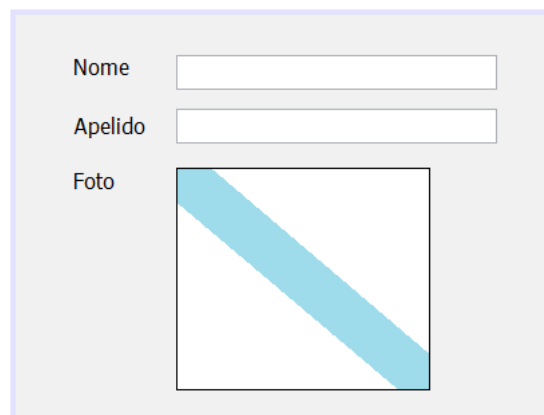
Cabe destacar unha etiqueta chamada `lblImaxe` que visualmente está representada pola caixa con borde negro na parte inferior do formulario. Como pódese observar nesta etiqueta, non é obrigatorio que unha etiqueta teña texto. Este pode ser establecido en tempo de execución a través do seu método `setText` en resposta a algunha acción realizada polo usuario. Tamén é destacable indicar que sobre este compoñente foi establecido un borde a través da súa propiedade `border`, por isto amosa un borde que rodea o seu su perímetro. Ben, unicamente fáltanos cargar a imaxe que queremos amosar dentro da etiqueta. Para elo teremos que modificar o valor da propiedade `icon` da etiqueta `lblFoto`. Empregando o entorno de programación accedemos ás propiedades da etiqueta `lblFoto` e pulsamos sobre o botón asociado á propiedade `icon`. Amósasenos a seguinte xanela:



A través desta xanela podemos seleccionar a localización da imaxe que queremos amosar dentro da nosa etiqueta. Podemos buscala dentro do proxecto (image within project), nun repositorio externo ao proxecto (external image) o podemos indicar que a etiqueta non amosará ningunha imaxe (no image). No exemplo sobre o que estamos a traballar a imaxe a amosar atópase nun paquete chamado `imaxes` dentro do noso proxecto e o nome da imaxe é `bandeira.png`. Polo tanto facemos esa selección e prememos OK para establecer o valor da propiedade `icon` da etiqueta `lblFoto`:



Unha vez establecida a propiedade icon para a etiqueta lblFoto, este pasa a ser o aspecto da xanela de deseño para o formulario:



Se executamos a aplicación este será o resultado:

