Construindo interfaces com PHP-GTK

Hora de construir

Enquanto o uso das classes de PHP-GTK cria interfaces rápidas, o Glade pode tornar a tarefa ainda mais fácil.

por Pablo Dall'Oglio

No primeiro artigo desta série[1], conhecemos o *PHP-GTK*, extensão da linguagem de programação *PHP*, que nos permite desenvolver aplicações gráficas *standalone*. Neste segundo artigo, iremos estudar as diversas formas de se construir uma interface.

Exemplo 1: Janela subdividida

```
01 <?php
02 // cria a janela
03 $janela = new GtkWindow;
04 $janela->set_size_request(300,200);
05 $janela->set_border_width(20);
06 $janela->set_title('Título da
➥Janela');
07
08 // cria boxes
09 $caixa_vert= new GtkVBox;
10 $caixa_horz= new GtkHBox;
11
12 // adiciona caixa horizontal na
➥janela
13 $janela->add($caixa_horz);
14
15 // adiciona elementos na caixa
➥horizontal
16 $caixa_horz->pack_start(new
➥GtkLabel('elemento1'));
17 $caixa_horz->pack_start($caixa_
➥vert);
18 $caixa_horz->pack_start(new
➥GtkLabel('elemento2'));
19
20 // adiciona elementos na caixa
➥vertical
21 $caixa_vert->pack_start(new
➥GtkLabel('elementoA'));
22 $caixa_vert->pack_start(new
➥GtkLabel('elementoB'));
23 $caixa_vert->pack_start(new
➥GtkLabel('elementoC'));
24
25 // exibe a janela
26 $janela->show_all();
27
28 Gtk::Main();
29 ?>
```

A biblioteca *GTK* é formada por um conjunto de classes, organizadas hierarquicamente e que disponibilizam uma interface totalmente orientada a objetos. Isso permite o reaproveitamento de código através de mecanismos como a herança. A grande maioria das classes descende da superclasse `GtkObject`. Toda classe-filha herda automaticamente o comportamento (métodos e propriedades) da classe pai. Para exemplificarmos, todas as classes descendentes de `GtkBin` são contêineres que podem conter um elemento em seu interior. Já as classes descendentes de `GtkBox` podem conter vários.

Empacotamento

A primeira forma de construirmos a interface gráfica é através do empacotamento de objetos, ou seja, colocando uns objetos dentro de outros. Para tal, é necessário entender quais objetos permitem que se adicione conteúdo (contêineres) e quais não.

Em princípio, objetos descendentes da classe `GtkContainer` podem conter outros elementos dentro de si. No **exemplo 1**, iremos construir uma janela (`GtkWindow`) e adicionar uma caixa horizontal em seu interior. Dentro da caixa horizontal, vamos adicionar três objetos, conforme a **figura 1**: dois rótulos de texto (`GtkLabel`), sendo um em cada extremidade, e uma caixa vertical no centro, sendo que essa caixa vertical contém outros três objetos do tipo `GtkLabel` (rótulos de texto).

Observe que podemos construir a interface recursivamente, colocando objetos dentro de objetos.

Posições Fixas

Quem considera complexo demais projetar o visual do aplicativo com base no empacotamento pode dispor da construção baseada em posições fixas. O GTK possui um contêiner chamado `GtkFixed`, que proporciona uma área na qual os objetos podem ser ancorados em coordenadas absolutas definidas em pixels. No **exemplo 2**, criamos um objeto chamado `$fixed_area`, que é justamente um objeto do tipo `GtkFixed` dentro da janela. A partir disso, colocamos vários objetos em seu interior (**figura 2**), sempre definindo a posição em termos de coluna e linha, por meio do método `put()`.

Glade

Até o momento, construímos o visual do aplicativo via programação. Essa forma de desenvolvimento melhora o desempenho do aplicativo, mas, por outro lado, não é tão produtiva para programadores acostumados com ambientes *RAD*. Para situações em que precisamos de maior agilidade no desenvolvimento visual do aplicativo, podemos utilizar o *Glade*.

Figura 1 Uma janela com três elementos dispostos horizontalmente.

Figura 2 Uma janela com múltiplos elementos, usando localização fixa.

O Glade é uma ferramenta criada especialmente para desenho visual de aplicativos GTK. Ele pode ser utilizado em conjunto com todas as linguagens que suportam o GTK.

Essa ferramenta disponibiliza ao programador uma paleta de componentes e uma janela de propriedades que propiciam a construção de interfaces por meio de recursos como clicar-e-arrastar, entre outros. O resultado final do arquivo salvo pelo Glade é um documento *XML* com a extensão `.glade`.

Exemplo 2: Janela com múltiplos elementos

```
01 <?php
02 // cria a janela
03 $janela = new GtkWindow;
04 $janela->set_size_request(300,200);
05 $janela->set_border_width(20);
06 $janela->set_title('Título da Janela');
07
08 // cria fixed
09 $fixed_area= new GtkFixed;
10
11 // adiciona área fixa na janela
12 $janela->add($fixed_area);
13
14 // instancia diversos objetos
15 $rotulo1 = new GtkLabel('elemento1');
16 $rotulo2 = new GtkLabel('elemento2');
17 $botao   = new GtkButton('clique aqui');
18 $check   = new GtkCheckButton('checkbutto
➥n');
19 $radio   = new GtkRadioButton(null,
➥'radiobutton');
20 $combo   = new GtkCombo;
21
22 // coloca objetos na área fixa
23 $fixed_area->put($rotulo1, 10,  10);
24 $fixed_area->put($rotulo2, 140, 10);
25 $fixed_area->put($botao,   10,  50);
26 $fixed_area->put($check,   120, 50);
27 $fixed_area->put($radio,   120, 80);
28 $fixed_area->put($combo,   0,  120);
29
30 // exibe a janela
31 $janela->show_all();
32
33 Gtk::Main();
34 ?>
```

No **exemplo 3**, utilizamos uma interface criada no Glade. Veja que para interpretar o documento Glade precisamos fazer uso da classe `GladeXML`. Essa classe retorna um objeto `$glade`, que disponibiliza acesso a todos os objetos contidos dentro do documento. Para obtermos tais objetos, precisamos executar o método `get_widget()`, que recebe o nome do objeto (exibido na janela de propriedades) e retorna o objeto correspondente, como se ele fosse instanciado naquele momento.

Exemplo 3: Uso de arquivo .glade

```
01 <?php
02 //realiza leitura do documento glade
03 $glade  = new GladeXML('exemplo.
➥glade');
04
05 //obtém a janela contida no glade
06 $janela = $glade->get_
➥widget('window1');
07
08 //obtém o campo para digitação do
➥nome da pessoa
09 $nome   = $glade->get_widget('nome_
➥pessoa');
10
11 //joga um texto dentro do campo
12 $nome->set_text('digite o nome
➥aqui...');
13
14 //exibe a janela
15 $janela->show_all();
16
17 Gtk::Main();
18 ?>
```

Conclusão

Neste artigo, apenas vimos como construir uma interface inanimada. No próximo artigo da série, vamos estudar como se dá a programação de eventos, que nos permitirá escrever funções que reagem às ações tomadas pelo usuário frente à aplicação. ■

Mais informações

[1] Pablo Dall'Oglio, "Sempre Nativo". Linux Magazine 37, dezembro de 2007, pág. 70.

[2] PHP-GTK Brasil: http://www.php-gtk.com.br

[3] Site do autor: http://www.pablo.blog.br

[4] Livro PHP-GTK: http://www.php-gtk.com.br/book

Sobre o autor

Pablo Dall'Oglio (pablo@php.net) é graduado em Análise de Sistemas, autor de um livro sobre PHP-GTK e programa em PHP-GTK desde sua criação em 2001. É membro da equipe de documentação e criador da comunidade brasileira de PHP-GTK. Atualmente, é diretor de tecnologia e proprietário da Adianti Solutions, onde atua como consultor de tecnologia e engenheiro de software.