Programação de sinais

Para conectar a interface gráfica de um programa ao restante do código, é preciso usar sinais.

por Pablo Dall'Oglio

Carsten Müller - www.sxc.hu

No artigo anterior, estudamos as diversas formas de se construir uma interface em *PHP-GTK*. Neste terceiro artigo, veremos como programar aplicativos para que estes respondam à interação com o usuário através da programação de sinais.

Widgets

Como vimos no artigo anterior, a biblioteca *GTK* é formada por um conjunto de classes organizadas hierarquicamente. A maioria dessas classes implementam *widgets*. "Widgets" é o termo utilizado para descrever um elemento gráfico de uma interface. Um widget pode ser um botão, um botão de rádio (*radiobutton*), um botão de marcação (*checkbutton*), uma imagem, uma lista ou mesmo uma janela (**figura 1**).

Contêineres

Alguns widgets podem conter outros em seu interior, como por exemplo a *janela*, o *frame*, o *notebook*, um *botão* ou uma *janela de rolagem*. Nesses casos, eles também são chamados de *containers* (contêineres). Um contêiner é utilizado para organizar a interface e distribuir seu conteúdo. Nem todo contêiner é visível, existem contêineres como as caixas vertical (*GtkVBox*) e horizontal (*GtkHBox*), além da tabela (*GtkTable*) que não são perceptíveis ao usuário, sendo que somente seu conteúdo é visível. A **figura 2** mostra alguns exemplos de contêiners.

Sinais e callbacks

Após criar uma interface, é necessário programar a interação do usuário com a aplicação. Diferentemente da Web, onde o fluxo de execução do aplicativo é baseado na requisição de páginas, em aplicações *standalone* como a GTK, a programação é baseada em sinais e eventos.

Um sinal é uma mensagem enviada por um widget sempre que o usuário interage com ele. Diferentes widgets emitem diferentes sinais sob diferentes circunstâncias. Pode parecer complexo, mas na verdade é bastante simples. Um botão, por exemplo, emite o si-

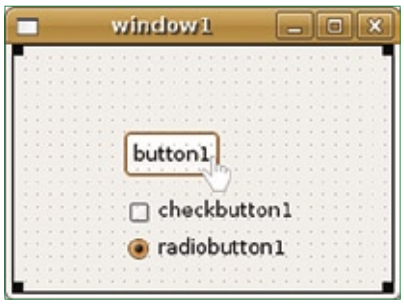

Figura 1 Elementos de uma janela.

nal `clicked` quando é clicado, `pressed` quando é pressionado e `released` quando é solto. Uma janela emite o sinal `destroy` quando é fechada, uma opção do menu emite o sinal `activate` quando é selecionada e uma lista emite o sinal `row-activated` quando o usuário clica duas vezes sobre um de seus itens.

Como o programador descobre essas palavras mágicas? Na documentação de cada widget. Se você acessar [1], verá a documentação de um botão, e lá constará uma relação dos seus sinais e sob quais circunstâncias eles são acionados.

Sempre que um sinal é emitido pela GTK, cabe ao programador definir o que será feito, criando uma função para responder à emissão deste sinal. As funções que são criadas para reagir a sinais são chamadas de *callbacks*. Uma callback pode ser uma função ou um método de um objeto como veremos nos exemplos a seguir.

Exemplo 1: Programa de cálculo de IMC

```
01 <?php
02 // calcula o índice de massa corpórea
03 function calcula_imc()
04 {
05   global $peso, $altura; // variáveis globais
06   $valor_peso   = $peso->get_text();
07   $valor_altura = $altura->get_text();
08   echo ($valor_peso/($valor_altura*$valor_altura)) . "\n";
09 }
10 // cria interface
11 $janela = new GtkWindow;
12 $peso   = new GtkEntry;
13 $altura = new GtkEntry;
14 $botao  = new GtkButton('calcula');
15
16 // conecta o sinal 'clicked' à função 'calcula_imc'
17 $botao->connect_simple('clicked', 'calcula_imc');
18
19 $vbox = new GtkVBox;
20 $vbox->add(new GtkLabel('Peso:'));
21 $vbox->add($peso);
22 $vbox->add(new GtkLabel('Altura:'));
23 $vbox->add($altura);
24 $vbox->add($botao);
25
26 $janela->add($vbox);
27 // exibe a janela
28 $janela->show_all();
29 Gtk::Main();
30 ?>
```

Programação estruturada

No **exemplo 1**, demonstramos a utilização de sinais e callbacks através de programação estruturada. Nele, construímos uma janela que pede ao usuário seu peso e altura, e quando este clica no botão *calcula*, o programa exibe o índice de massa corpórea no console. Note que os comandos `echo` e `print`, que na Web exibem o resultado em uma página HTML, na GTK exibem os resultados na linha de comando na qual o programa foi executado. A GTK possui classes para exibição de diálogos com o usuário que são mais apropriadas para exibir mensagens.

Figura 2 Contêineres são elementos especiais que podem conter outros elementos em seu interior.

Veja que no **exemplo 1** criamos uma janela (**figura 3**) e alguns objetos (`$peso`, `$altura`, `$botao`) e os adicionamos à janela por meio de uma caixa vertical (`GtkVBox`). A parte mais importante desse exemplo é o momento em que conectamos o sinal `clicked` do botão à função `calcula_imc()`. Com isso, ordenamos ao GTK que execute tal função sempre que o botão for clicado. Como os objetos criados (`$peso`, `$altura`) não existem dentro da função `calcula_imc()`, tivemos de utilizar variáveis globais para tornar as variáveis visíveis em todo o programa. Esse recurso deve ser evitado ao máximo, e foi utilizado aqui somente por motivos didáticos. O cálculo do índice é realizado dividindo-se o peso pelo quadrado da altura. Para obter os valores digitados em um campo de digitação do tipo `GtkEntry` (entrada), utilizamos o método `get_text()`.

Programação orientada a objetos

No **exemplo 1**, vimos como construir uma janela utilizando a programação estruturada. No próximo exemplo, vamos escrever um programa muito parecido, mas dessa vez utilizando a orientação a objetos.

A utilização da orientação a objetos nos dá maior flexibilidade e modularidade, permitindo reutilizar muito mais facilmente estruturas lógicas já existentes. O próprio GTK permite que criemos novas classes baseadas nas suas.

No **exemplo 2**, criamos uma janela para cálculo da média entre dois números. Veja que essa janela (**figura 4**), chamada `CalculaMedia`, é filha de `GtkWindow`, uma classe nativa do GTK que implementa uma janela.

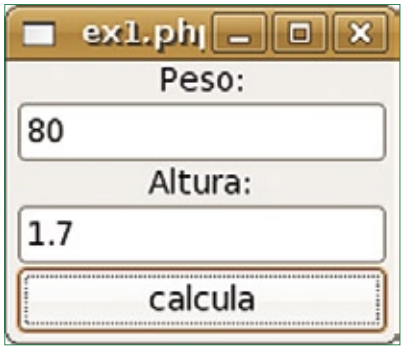

Figura 3 Janela do programa que exibe o índice de massa corpórea.

Assim, por meio da utilização de herança, reaproveitamos toda a estrutura já existente de uma janela e a especializamos, adicionando alguns comportamentos específicos.

No **exemplo 2**, usamos dois objetos (`$valor1` e `$valor2`), além de um botão para calcular o resultado (`$botao`) e um rótulo de texto (`$result`) que será utilizado para exibir o resultado do cálculo. Quando o usuário clicar no botão *calcula*, o método `calcula_media()` será executado e o resultado da média entre os dois valores será calculado e exibido no lugar do rótulo de texto `$result`.

Nessa classe, optamos por definir todo o visual da janela no seu método construtor (`construct`). Nesse caso, não precisamos mais utilizar variáveis globais. No lugar delas, optamos por declarar os objetos que precisaríamos acessar, como atributos dessa classe (`private $valor1`, por exemplo). Dessa forma, eles são acessíveis ao longo de toda a classe.

Note que no programa do **exemplo 2**, a forma de conectar o sinal `clicked` do botão a uma função se alterou. No lugar de especificarmos somente o nome de uma função no método `connect_simple()`, passamos como parâmeto um vetor formado por duas posições: a primeira indica um objeto e a segunda o nome de um método a ser executado.

Nesse programa, no lugar de exibir o resultado do cálculo no console do sistema, estamos jogando-o de volta à tela, por meio de um rótulo de texto localizado abaixo do botão calcula (`$result`).

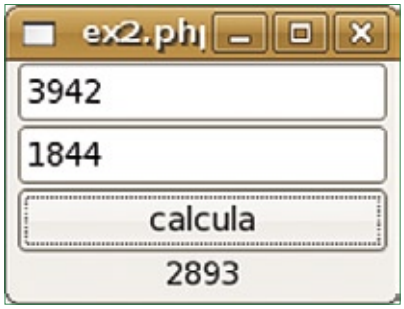

Figura 4 Janela do programa que calcula a média entre dois números.

Exemplo 2: Média entre dois números

```
01 <?php
02 class CalculaMedia extends GtkWindow
03 {
04   private $valor1;
05   private $valor2;
06   private $result;
07   public function __construct()
08   {
09     parent::__construct(); // cria a janela
10     // cria os campos para digitação
11     $this->valor1 = new GtkEntry;
12     $this->valor2 = new GtkEntry;
13     $this->result = new GtkLabel('resultado');
14     $botao  = new GtkButton('calcula');
15     // conecta o sinal 'clicked' ao método 'calcula_media'
16     $botao->connect_simple('clicked', array($this, 'calcula_media'));
17     // coloca os campos em uma caixa vertical
18     $vbox = new GtkVBox;
19     $vbox->add($this->valor1);
20     $vbox->add($this->valor2);
21     $vbox->add($botao);
22     $vbox->add($this->result);
23     parent::add($vbox);
24     // exibe a janela
25     parent::show_all();
26   }
27   // calcula a média entre dois números
28   function calcula_media()
29   {
30     $media = ($this->valor1->get_text() +
31           $this->valor2->get_text()) / 2;
32     $this->result->set_text($media);
33   }
34 }
35 new CalculaMedia;
36 Gtk::Main();
37 ?>
```

Conclusão

Neste artigo vimos como responder às interações do usuário com a interface. No próximo artigo da série, iremos estudar como se dá a integração com bancos de dados utilizando o *Sqlite*, que é um banco de dados *standalone* e de fácil distribuição. ■

Mais informações

[1] Widget de botão: http://www.php-gtk.com.br/gtkbutton

[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 dos softwares Agata Report e Tulip e também de dois livros sobre PHP, além de manter a comunidade brasileira de PHP-GTK. É membro da equipe de documentação e criador da comunidade brasileira de PHP-GTK. Atualmente, é diretor de tecnologia e proprietário da Adianti Solutions (www.adianti.com.br).