por Pablo Dall'Oglio

Gerando planilhas em Excel com PHP-GTK

Neste artigo iremos construir uma ferramenta para lançamento de despesas financeiras cujo objetivo é criar uma planilha no formato Excel.

O PHP é uma linguagem incrível em relação à quantidade de bibliotecas de terceiros que gravitam em torno de sua órbita. Temos bibliotecas para os mais variados fins, como envio de emails, manipulação de imagens, geração de documentos PDF, dentre outros.

O PHP possui um grande repositório de classes chamado PEAR (PHP Extension and Application Repository), como muitos de vocês já devem saber. Este grande repositório orientado a objeto é constantemente fruto de artigos em revistas e sites sobre PHP. Neste artigo em específico, vamos provar que podemos utilizar praticamente qualquer bibliteca comumente utilizada no ambiente Web no PHP-GTK. Para isso, escolhemos a biblioteca Spreadsheet Excel Writer, que é um pacote responsável pela geração de arquivos em formato Excel. Para mostrar seu funcionamento em conjunto com o PHP-GTK, iremos construir um programa para lançamentos de despesas, que irá inserindo estes lançamentos em uma listagem para posterior geração do arquivo em formato Excel.

1. Listas e Árvores

Listas e árvores no PHP-GTK são implementados através da classe GtkTreeView, que pode ser utilizada para exibir tanto árvores quanto listas.

Figura 1 – Componente GtkTreeView

O conceito mais importante por trás de uma GtkTreeView é o fato dela implementar o padrão MVC (Model, View, Controller), ou seja, a separação entre os dados e a forma pela qual estes são exibidos em tela. Os dados, sejam eles números, textos ou imagens são armazenados em um modelo, que pode ser GtkListStore (para armazenar listas) ou GtkTreeStore (para armazenar árvores). O modelo de dados é, então, atribuído a alguma visualização (GtkTreeView). Sempre que o modelo de dados é alterado, automaticamente sua exibição é atualizada em tela, por sua visualização (GtkTreeView).

A camada Controller contém objetos que capturam ações derivadas de interação do usuário, coordenando assim, objetos das camadas Model e View para fornecer a resposta adequada ao usuário.

2. Spreadsheet Excel Writer

Como dissemos anteriormente, a biblioteca Spreadsheet Excel Writer permite a criação de planilhas no formato Excel. Ela suporta fórmulas, imagens e formatação de textos e células.

Assim como toda biblioteca que faz parte do PEAR, a SpreadSheet Excel Writer deve ser instalada através do seguinte comando:

```
#pear install SpreadSheet_Excel_Writer-0.9.1
```

Após isto, os seus arquivos devem ser disponibilizados dentro da pasta lib de sua instalação do PHP.

3. O programa

Para começar, iremos estender a classe GtkWindow e criar toda a interface em seu método construtor. A nossa janela de lançamentos terá basicamente uma caixa vertical (GtkVBox) contendo uma série de caixas horizontais (GtkHBox).

Em cada caixa horizontal, teremos um par de rótulos (GtkLabel) e um campo de entrada de dados (GtkEntry). Ao final, criamos um botão adicionar (STOCK_ADD), cujo objetivo é coletar os dados digitados pelo usuário nestes campos e adicionar em uma listagem (objeto GtkTreeView). Este botão está conectado ao método **onAddLinha**

(). Isto significa que sempre que o botão for clicado, este método será executado. Também temos um botão de salvar (STOCK_SAVE), cujo objetivo é coletar todos os lançamentos da listagem e gerar a planilha Excel a partir deles.

Neste método construtor, ainda definimos os tamanhos dos campos por meio do método **set_size_request()** e executamos o método **createTreeView()**, cuja responsabilidade é declarar nossa listagem de lançamentos e suas respectivas colunas. Vários pontos deste programa foram suprimidos em virtude do espaço aqui reduzido, mas poderão ser encontrados no site indicado ao final do artigo.

```
<?php
/*
 * Exportar Excel
 * Planilha de gastos com exportação de excel
 * Adianti Solutions (www.adianti.com.br)
 */
class ExportarExcel extends GtkWindow
{
private $list;       // apresentação dos dados
private $model;      // modelo de dados

private $descricao; // campo de descrição
private $data;       // campo de data
private $valor;      // campo de valor
private $operacao;  // campo de operação

/*
 * Método construtor
 * Instancia a janela e cria toda interface
 */
function __construct()
{
    // cria a janela
    parent::__construct();
    parent::set_border_width(4);
    parent::set_size_request(470,300);

    // cria uma caixa vertical
    $vbox = new GtkVBox;

    // cria rótulos de texto
    $label1 = new GtkLabel('Descrição : ');
    $label2 = new GtkLabel('Data: ');
    ...
    // define o tamanho dos rótulos
    $label1->set_size_request(200,-1);
    $label2->set_size_request(200,-1);
    ...
    // cria os campos de entrada de dados
    $this->descricao    = new GtkEntry;
    $this->data         = new GtkEntry;
    $this->valor        = new GtkEntry;
    $this->operacao     = new GtkEntry;

    // define os tamanhos dos campos
    $this->descricao->set_size_request(240...
    $this->data->set_size_request(80,-1);
    ...
    // cria caixas horizontais
    $hbox1 = new GtkHBox;
    $hbox2 = new GtkHBox;
    $hbox3 = new GtkHBox;
    $hbox4 = new GtkHBox;

    // adiciona os labels em suas caixas
    $hbox1->pack_start($label1, false, false);
    $hbox2->pack_start($label2, false, false);
    ...
    // adiciona os campos em suas caixas
    $hbox1->pack_start($this->descricao...);
    $hbox2->pack_start($this->data...);
    $hbox3->pack_start($this->valor...);
    $hbox4->pack_start($this->operacao...);

    // adiciona as caixas na caixa vertical
    $vbox->pack_start($hbox1, false, false);
    $vbox->pack_start($hbox2, false, false);
    ...
    // cria um botão de adicionar
    $button =
    GtkButton::new_from_stock(Gtk::STOCK_ADD);
    // define a ação do botão
    $button->connect_simple('clicked',
    array($this, 'onAddLinha'));

    // cria o objeto TreeView
    $this->createTreeView();

    // adiciona o TreeView na caixa vertical
    $vbox->pack_start($this->list);

    // cria um botão de salvar
    $button =
    GtkButton::new_from_stock(Gtk::STOCK_SAVE)
    // define a ação do botão
    $button->connect_simple('clicked',
    array($this, 'onExportar'));

    // adiciona a caixa vertical na janela
    parent::add($vbox);
}
```

O método **createTreeView()** será executado a partir do método construtor e seu objetivo é instanciar o objeto GtkTreeView, para implementar a listagem de lançamentos. Esta listagem armazenará os dados em um objeto GtkListStore (propriedade list), que terá quatro colunas do tipo string.

```
function createTreeView()
{
    // instancia treeview
    $this->list = new GtkTreeView;

    // cria modelo
    $this->model = new GtkListStore(
    Gtk::TYPE_STRING, Gtk::TYPE_STRING...);
    // define o modelo da treview
    $this->list->set_model($this->model);

    // cria quatro colunas
    $column1 = new GtkTreeViewColumn('Desc.');
    $column2 = new GtkTreeViewColumn('Data');
    ...
    // define as larguras
    $cell_renderer1->set_property('width'...);
    $cell_renderer2->set_property('width'...);
    ...
    // adiciona as colunas na lista
    $this->list->append_column($column1);
    $this->list->append_column($column2);
    ...
}
```

Sempre que o usuário clicar no botão “Adicionar”, o método **onAddLinha()** será executado. O objetivo deste método é obter os valores digitados pelo usuário nos

campos de entrada de dados que são armazenados em propriedades deste objeto (descrição, data), através do método **get_text()** da classe GtkEntry.

Quando obtemos os valores destas variáveis, adicionamos eles ao modelo de dados da listagem (GtkTreeView) através do método **append()**, passando um array com o conteúdo da linha. Posteriormente, limpamos os conteúdos dos campos com o método set_text ('') e posicionamos o foco do cursor no campo para descrição.

```
function onAddLinha()
{
    // obtém os valores digitados pelo usuário
    $descricao = $this->descricao->get_text();
    $data      = $this->data->get_text();
    $valor     = $this->valor->get_text();
    $operacao  = $this->operacao->get_text();

    // adiciona na lista
    $this->model->append(array($descricao,
                $data, $valor, $operacao));

    // limpa os conteúdos digitados
    $this->descricao->set_text('');
    $this->data->set_text('');
    ...
    // posiciona o foco no campo descrição
    parent::set_focus($this->descricao);
}
```

O método **onExportar()** será executado sempre que o usuário clicar no botão "Salvar". O primeiro passo é incluir a biblioteca SpreadSheet Excel Writer para disponibilizar esta classe para a aplicação. Após, abrimos um diálogo de salvar arquivos para coletar o nome e o caminho onde o usuário deseja salvar a planilha.

```
function onExportar()
{
    // inclui a biblioteca SpreadsheetWriter
    require_once "Spreadsheet/Excel/Writer...'

    // abre um diálogo de salvar arquivo
    $dialog = new GtkFileChooserDialog...;

    // exibe diálogo
    $response = $dialog->run();

    // Verifica resposta do usuário
    if ($response == Gtk::RESPONSE_OK)
    {
        // obtém o nome do arquivo
        $filename = $dialog->get_filename();
    }
    // destrói diálogo
    $dialog->destroy();
```

Depois de coletar o nome do arquivo, instanciamos um objeto da classe Spreadsheet_Excel_Writer, responsável por fornecer os métodos que nos permitem gerar as planilhas em formato Excel.

```
// cria uma planilha Excel
    $xls = new Spreadsheet_Excel_Writer
($filename);
```

O funcionamento desta classe é bastante simples. Primeiro definimos um conjunto de "estilos" a serem utilizados para as células. Cada estilo define características como tamanho, cor e fonte. Posteriormente, utilizamos estes estilos pelo método **write()**.

```
// define um formato para o título
    $titleFormat = $xls->addFormat();
    $titleFormat->setFontFamily('Helvetica');
    $titleFormat->setSize('14');
    $titleFormat->setColor('black');
    ...
    // define um formato para as colunas
    $columnFormat = $xls->addFormat();
    $columnFormat->setFontFamily('Helvetica');
    $columnFormat->setSize('12');
    $columnFormat->setColor('black');
    ...
    // define um formato para os dados
    $dataFormat = $xls->addFormat();
    $dataFormat->setFontFamily('Courier');
    $dataFormat->setSize('12');
    $dataFormat->setColor('black');
    ...
```

Após definirmos alguns estilos, utilizamos o método **addWorkSheet()** para criar uma aba, ou uma planilha de trabalho dentro do nosso arquivo. Veja na parte inferior da figura 3, o nome "PHP Magazine".

Após isto, escrevemos os cabeçalhos da planilha através do método **write()**, indicando o endereço de cada célula, através de coordenadas linha e coluna.

```
    // Adiciona uma aba à planilha
    $sheet = $xls->addWorksheet('PHP Magazi-
ne');

    // escreve nas células (linha, coluna)
    $sheet->write(0,1,'Caixa', $titleFormat);
    $sheet->write(1,0,'Descrição'...);
    $sheet->write(1,1,'Data', $columnFormat);
    $sheet->write(1,2,'Valor'...);
    $sheet->write(1,3,'Operação'...);
```

Depois de escrevermos os cabeçalhos da planilha, percorremos os dados da listagem (objeto GtkTreeView), coletamos célula a célula, através do método **get_value()** e escrevemos estes valores dentro da planilha através do método **write()**.

```
// percorre o modelo de dados
    $i = 2;
    $iter = $this->model->get_iter_first();

    while ($iter)
    {
        // obtém os valores do modelo de dados
        ...
        $valor      =
        $this->model->get_value($iter, 2);
        $operacao   =
        $this->model->get_value($iter, 3);

        // escreve os dados na planilha
        $sheet->write($i,2,$valor...);
        $sheet->write($i,3,$operacao...);

        // pega o próximo iterador
```

```
            $iter =
            $this->model->iter_next($iter);
            $i ++;
        }
```

Ao final, fechamos o objeto planilha e abrimos um diálogo de sucesso com a mensagem Planilha Gerada!!

```
// fecha a planilha
    $xls->close();
    // abre um diálogo de mensagem de sucesso
    $dialog = new GtkMessageDialog
(...'Planilha gerada !!!');
    $response = $dialog->run();
    // fecha o diálogo
    $dialog->destroy();
}
}
```

Aqui é onde começa o nosso programa na verdade, a classe ExportaExcel é instanciada

```
// instancia a classe ExportaExcel
$janela = new ExportarExcel;
$janela->show_all();
Gtk::Main();
?>
```

Aqui você confere a tela principal do nosso software de controle de lançamentos financeiros.

Figura 2 – Tela do gerador de planilhas

Nesta tela seguinte, você confere a planilha eletrônica gerada no formato Excel.

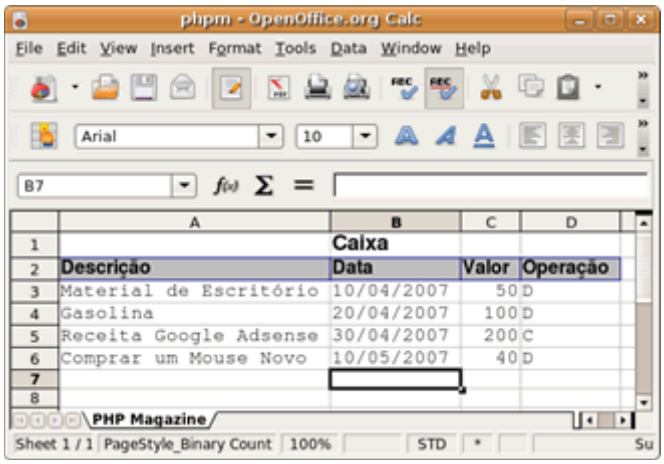

Figura 3 – Planilha gerada

Considerações finais

Neste artigo, com menos de 300 linhas de código, escrevemos um simples programa para lançamentos financeiros com geração de planilhas Excel em PHP-GTK. Como o programa ficou um pouco grande para o formato da revista, optamos por cortar alguns pedaços, substituindo-os por "..." por motivos didáticos. Você pode fazer download da aplicação completa no site da pSheet, em http://psheet.php-gtk.com.br

Referências e links sugeridos

[PHP-GTK Brasil] – http://www.php-gtk.com.br
[Livro PHP-GTK] – http://www.php-gtk.com.br/book
[Site do Autor] – http://www.pablo.blog.br
[Site da Planilha] –http://psheet.php-gtk.com.br
[Site do PEAR] –http://pear.php.net

Pablo Dall'Oglio - pablo@dalloglio.net
Pablo Dall'Oglio é formado em Análise de Sistemas pela UNISINOS. Autor do livro sobre PHP-GTK pela Novatec Editora. Programa em PHP-GTK desde sua criação em 2001. É membro do time de documentação e criador da comunidade brasileira de PHP-GTK (www.php-gtk.com.br). Atualmente, é diretor de tecnologia e proprietário da Adianti Solutions (www.adianti.com.br), onde atua como consultor de tecnologia e engenheiro de software. Pode ser contatado pelo e-mail pablo@php.net.