Desenvolvimento acelerado de aplicativos com Ruby e Glade

Interface preciosa


Henrik Gerold Vogel - www.pixelio.de

O desenvolvimento de aplicativos gráficos não deveria ser complicado. Veja como simplificar esse processo com Ruby e o utilitário Glade.

por Ryan Paul

Mesmo os desenvolvedores mais experientes podem ficar cansados do ciclo de edição-compilação-e-teste. Felizmente, algumas ferramentas de código aberto eliminam o peso desse processo, permitindo que os programadores voltem sua atenção à funcionalidade, em vez de todos os detalhes arbitrários de implementação.

Com as linguagem de programação dinâmicas e utilitários de criação de interface versáteis, até mesmo programadores relativamente inexperientes conseguem construir aplicativos complexos com pouco esforço. Uma combinação de ferramentas bastante promissora é composta pela linguagem Ruby e o sistema de desenvolvimento de interfaces Glade.

A versátil e expressiva linguagem Ruby permite que os desenvolvedores escrevam programas que executem mais tarefas com menos código. O Glade traz o poder do toolkit GTK+, possibilitando a criação fácil de interfaces gráficas para aplicativos em Ruby.

Neste artigo, serão mostrados todas as etapas para a criação de um aplicativo com Ruby e Glade. Mostraremos ambas as ferramentas em ação, enquanto criamos um aplicativo capturador de telas de exemplo, o Captura.

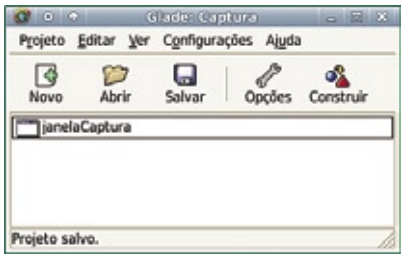

Figura 1 A janela do projeto lista todas as janelas de nível mais alto e diálogos incluídos no atual projeto do *Glade*.

A sintaxe orientada a objetos concisa e expressiva do Ruby leva a um desenvolvimento acelerado. Programas em Ruby bem escritos são fáceis de ler, entender e manter. Há uma grande variedade de bibliotecas de programação disponíveis para essa linguagem, o que a torna muito usada na tarefa de "colar" funcionalidades de fontes externas. A escalabilidade do Ruby ainda oferece um caminho suave, que facilita a evolução de scripts de linha de comando para utilitários gráficos com menus, barras de ferramentas e caixas de texto.

O Glade é um sistema de desenvolvimento de interfaces baseado no toolkit GTK+. Ele consiste em um programa de construção de interfaces e um conjunto de bibliotecas para facilitar o carregamento e a manipulação de interfaces.

O construtor de interfaces suporta a criação de interfaces com o mouse e gera simples arquivos XML de descrição de interface. Os desenvolvedores usam as bibliotecas do Glade para tratar os arquivos XML e gerar interfaces GTK na memória durante a própria execução. O programa então consegue exibir e manipular essas interfaces como se houvessem sido feitas manualmente em GTK.

O Glade oferece várias vantagens únicas em relação às outras ferramentas de criação de interfaces. Devido ao uso que ele faz do carregamento em tempo de execução, em vez de gerar código estático, ele não é específico para uma linguagem, o que significa que arquivos do Glade podem ser usados com qualquer linguagem de programação para a qual haja ligações (bindings) para o Glade. O carregamento em tempo de execução também reduz o tempo necessário para os testes, já que não são necessários passos adicionais entre a modificação da interface e a execução do programa que a utiliza.

Primeiros passos

O utilitário de construção de interfaces Glade compreende quatro janelas flutuantes. A janela de projeto lista os formulários associados ao projeto atual, enquanto a janela Paleta oferece acesso a todos os widgets e controles GTK acessíveis no Glade. A janela de propriedades mostra vários atributos do widget selecionado e a janela da árvore de widgets apresenta uma hierarquia de todos os controles utilizados no projeto atual. Quando o Glade é iniciado, todas as janelas estão vazias ou desativadas, até que seja criado um projeto.

Para iniciar um novo projeto, selecione Novo no menu Projeto na janela de projetos (**figura 1**). Ao iniciar um novo projeto, os usuários precisam optar entre GTK e Gnome. Apesar de os projetos para Gnome terem acesso a um conjunto maior de widgets, os projetos em GTK são mais portáveis e possuem menos dependências. Neste

tutorial será mostrada a criação de um projeto GTK.

Após criar um novo projeto, a janela Paleta (**figura 2**) ficará acessível. Ela inclui botões para a categoria e o widget. Cada categoria contém alguns widgets que podem ser incluídos nos programas do Glade. O usuário pode alternar entre categorias clicando nos respectivos botões. A categoria Básico inclui vários widgets freqüentemente usados em aplicativos simples. Adicional contém widgets um pouco mais obscuros, mas ainda usados com freqüência. A categoria Obsoleto inclui widgets que não são mais suportados, mas ainda ocasionalmente usados por aplicativos legados. Para selecionar um widget, clique em um dos botões em alguma categoria.

Layout

Para criar nosso aplicativo de captura de tela, vamos começar projetando a interface com o Glade, e depois usaremos o Ruby para associar as ações aos eventos do programa.

Para criar uma janela para o utilitário de captura de tela, selecione o widget Janela na categoria Básico da Paleta. Ao clicar nesse botão, uma nova janela será aberta, e a janela Propriedades (**figura 3**) exibirá os atributos da nova janela. Com o nome padrão de window1, essa janela será listada automaticamente na janela do projeto e também em sua árvore de widgets. Quando novas janelas do Glade são criadas, seu fundo aparece cinza e quadriculado para mostrar que estão vazias.

O primeiro item na janela de propriedades é o nome do widget. Os programas em Ruby que carregam interfaces gráficas do Glade conseguem acessar e manipular widgets individuais dentro da interface do usuário, referindo-se aos nomes dos widgets. Como resultado, é importante especificar nomes com significado real para os widgets que o programa precisará manipular. Para mudar o valor da propriedade Nome, é necessário apenas digitar o novo nome na caixa de texto de propriedade de Nome. Para o utilitário de captura de tela, especifique a propriedade Nome da janela como janelaCaptura. A propriedade Título é usada para especificar o texto que aparecerá na barra de título da janela. Para este exemplo, vamos alterar o valor dessa propriedade para Ferramenta de Captura de Tela. Apesar de muitas outras propriedades de janelas estarem disponíveis, este artigo tratará apenas das propriedades relevantes para o aplicativo de exemplo. Os usuários podem obter mais informações a respeito de outras propriedades de widgets na documentação da GTK.

Em aplicativos GTK, os widgets são empilhados em caixas aninháveis, que podem ser verticais, horizontais ou em grade. Essa técnica de layout é chamada de modelo de caixa (box model). Para determinar quais tipos de caixas devem ser usadas, primeiro é necessário considerar a estrutura da interface do aplicativo. Nossa ferramenta Captura possui uma barra de menus, outra de ferramentas, um widget de seleção de arquivo e um painel de pré-visualização. Os widgets serão empilhados verticalmente, e o painel de pré-visualização precisará crescer quando o usuário redimensionar a janela. Para acomodar esse layout, o programa usará uma caixa vertical com quatro linhas.

Para acrescentar à janela uma caixa vertical, basta selecionar o widget de caixa vertical na categoria Básico da paleta do Glade. Após clicar no botão da caixa vertical, clique em qualquer ponto da janela vazia do aplicativo, e crie quatro linhas no diálogo será mostrado, pois nosso programa possuirá quatro widgets principais.

Depois de adicionar a caixa vertical, o espaço em branco na janela do utilitário será particionado em quatro retângulos separados, e a lista da árvore de widgets da janela incluirá a caixa vertical.

Para adicionar ou remover linhas da caixa vertical, pode-se utilizar sua propriedade Tamanho, e para especificar a quantidade de espaço em branco que o aplicativo deve mostrar entre cada par de linhas, pode-se usar a propriedade Espaçamento. Para modificar novamente as propriedades da janela, selecione a janela na árvore de widgets. É possível usar a árvore de widgets para selecionar e manipular qualquer widget do programa atual.

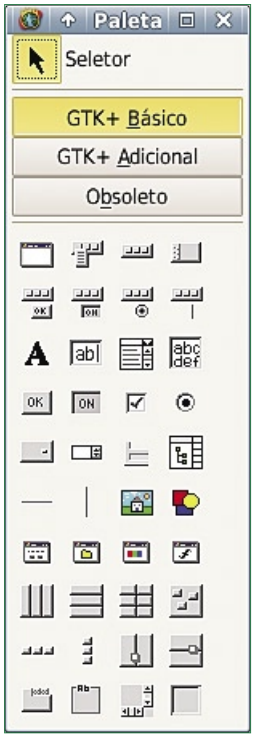

Figura 2 A janela da *Paleta* oferece acesso a todos os *widgets* e componentes que podem ser incluídos em projetos *Glade*.

Barra de menu

Selecione o widget da barra de menu a partir da categoria Básico da paleta do Glade. Em seguida, clique dentro da linha superior da caixa vertical. Aparecerá uma barra de menu padrão, com os menus Arquivo, Editar, Exibir e Ajuda. Será necessário modificar os menus para

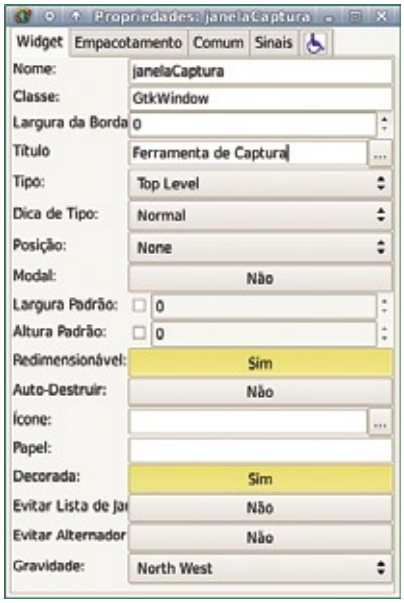

Figura 3 A janela de *Propriedades* permite que os desenvolvedores customizem os atributos de *widgets*.

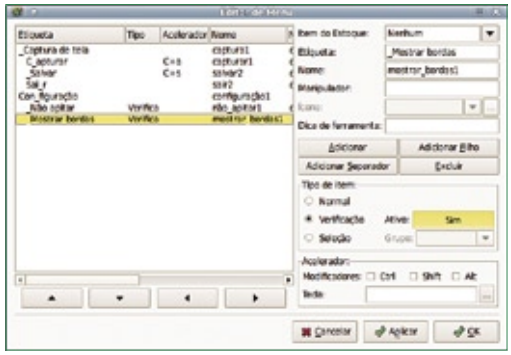

Figura 4 O Editor de Menu permite a personalização dos menus do aplicativo.

melhor se adequarem ao programa de exemplo, então é preciso clicar com o botão direito sobre a barra de menu e selecionar Editar (**figura 4**). O lado esquerdo do Editor de Menu contém a árvore de itens do menu, e o lado direito exibe as propriedades do menu selecionado, com opções de alteração do mesmo.

Antes de acrescentar itens de menu propriamente ditos, é necessário apagar os já existentes. Feito isso, clique em Adicionar para criar um novo menu.

A propriedade Etiqueta especifica o texto que será exibido no menu. O primeiro item de nossa barra será o menu Captura. Em diversos programas, cada item do menu possui uma letra sublinhada para indicar o que o usuário pode digitar junto à tecla **[Alt]** para ativar o referido item. A GTK trata a funcionalidade do teclado automaticamente, e é possível usar um traço baixo (_) na etiqueta do widget para determinar qual tecla a GTK deve associar ao item de menu. Digite um traço baixo logo antes da letra C de Captura para que os usuários consigam abrir esse menu através do atalho **[Alt]** + **[C]**.

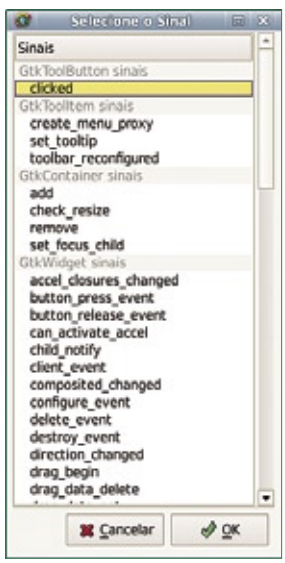

Figura 5 O diálogo de seleção de sinais é usado para selecionar um sinal de evento a ser associado a um manipulador.

Para associar uma função ao evento de seleção do menu, usa-se a propriedade Manipulador dos itens de menu. O valor dessa propriedade é o nome do método que o aplicativo invocará no código-fonte quando o usuário selecionar o respectivo item de menu. Como Configurações é um menu, e não um item de menu, sua propriedade Manipulador deve ficar vazia. Apague o texto desse campo também no menu Imagem.

Em seguida, vamos acrescentar itens ao menu Captura de tela. Selecione o item Captura de tela na árvore de itens de menu e clique em Adicionar filho. Esse novo item de menu permitirá que os usuários iniciem a captura de tela. Especifique a Etiqueta como `_Capture` e o Manipulador como `ao_capturar`. Quando for escrito o código-fonte do aplicativo, o método `ao_capturar` será implementado para realizar a captura da tela. Como os usuários provavelmente usarão com freqüência o recurso de captura de tela, é interessante atribuir-lhe um atalho de teclado, chamado de Acelerador. Para isso, selecione Captura de tela na árvore de itens e especifique os Aceleradores **[Ctrl]** e **[C]**.

Em seguida, clique no botão Adicionar para incluir esse item no menu. Digite `_Salvar` na Etiqueta, selecione o Ícone _Salvar e digite `ao_salvar` no Manipulador, completando com o Acelerador **[Ctrl]** + **[S]**. O último item do menu Captura de tela permitirá que o usuário feche o programa. Clique no botão Adicionar, especifique a etiqueta como `Sai_r`, o Manipulador como `ao_sair` e escolha o ícone respectivo.

Nosso programa de captura de tela também terá um menu de configurações. Clique no botão Adicionar para acrescentar um novo item de menu. Note que é necessário tomar cuidado com o nível hierárquico do novo item. Lembre-se de que o Adicionar Filho acrescenta um submenu ao item selecionado, enquanto o Adicionar cria um novo item no nível atual. Agora, especifique `Con_figuração` na Etiqueta e apague o valor do Manipulador. Os itens no menu Configuração serão usados para o usuário alterar o comportamento do programa. Em vez de usar manipuladores, os itens desse menu terão caixas do tipo checkbox que o usuário pode marcar ou desmarcar. O estado dessas caixas será usado durante a execução para determinar como o programa opera.

Para adicionar o primeiro item ao menu Configuração, clique no botão Adicionar Filho. Esse item permite que o usuário especifique se deseja que o programa apite após completar com sucesso uma captura de tela. Digite na Etiqueta `Não _Apitar` e apague o Manipulador.

Para transformar esse item em uma checkbox, selecione Verificação no grupo Tipo de item. Para fazer com que o programa por padrão não emita um apito, selecione Sim para o botão Ativo. Como o programa acessará o valor da checkbox em tempo de execução, o item de menu Não Apitar precisa ter um valor significativo na propriedade Nome. Digite `marcarNaoApitar` no Nome do item Não Apitar.

Queremos permitir que o usuário configure se a captura deve incluir as bordas das janelas. Para criar uma opção nesse sentido, clique no botão Adicionar, digite `_Mostrar bordas` e apague o Manipulador. Para garantir que as capturas incluam as bordas por padrão, deixe o Tipo de item em Verificar e certifique-se de que o valor da propriedade Ativo seja Sim. Depois, especifique o valor `marcarMostrarBordas` na propriedade Nome. Ao terminar a criação dos menus, clique em OK.

Barra de ferramentas

Criemos agora uma barra de ferramentas na caixa vertical. Selecione o widget de barra de ferramentas a partir da categoria Básico na Paleta. Então clique dentro da segunda partição da caixa vertical, logo abaixo do menu. O Glade exibirá uma caixa de diálo-

go que pergunta quantos itens devem ser incluídos na barra de ferramentas. Nossa barra terá três botões, que executam as mesmas operações que os itens do menu Captura de tela, assim! é necessário especificar o valor de `3` e clicar em OK. Muitos widgets GTK que podem ser colocados numa janela também têm a possibilidade de serem inseridos numa barra de ferramentas. Porém, nesse caso, usaremos os botões da barra de ferramentas, então o melhor é selecionar o botão de widget da barra de ferramentas a partir da categoria Básico da Paleta. Em seguida, clique no primeiro espaço vazio da barra de ferramentas para criar um novo botão. Preencha da mesma forma os outros espaços vazios.

Neste momento, é necessário especificar os atributos dos novos botões. Clique no primeiro deles, na janela de nosso programa, e depois selecione a aba Widget da janela de propriedades. Esse botão realizará a função de captura. Digite `Capturar` no campo Etiqueta e depois selecione um ícone para a propriedade Ícone. Depois, é necessário associar esse botão ao método de captura. Com o primeiro botão da barra de ferramentas selecionado na janela de nosso programa, selecione a aba Sinais.

No GTK, os sinais são mensagens emitidas mediante a ocorrência de um evento. Nesse caso, queremos que o programa chame o método `ao_capturar`, então é preciso detectar o evento de clique do botão. Para abrir o diálogo de seleção de sinal, clique no pequeno botão à direita da caixa de texto Sinal (**figura 5**). O diálogo de seleção de sinal lista todos aqueles que podem ser emitidos pelo widget selecionado, incluindo os que são herdados de outros widgets. O nome do sinal selecionado aparecerá na caixa de texto de propriedade, em Sinais. Como o botão Capturar fará a mesma coisa que o item de menu de mesmo nome, os dois devem acionar o mesmo manipulador: o método `ao_capturar`. Especifique o valor da propriedade Manipulador como `ao_capturar` e depois adicione o botão à janela Propriedades.

Selecione o segundo botão da barra de ferramentas na janela do programa. Por estar associado à função de gravação da imagem capturada, usada com freqüência, é possível usar o botão GTK padrão, em vez de selecionar manualmente a etiqueta e o ícone.

Selecione Salvar na caixa combo das propriedades do botão GTK, no diálogo de propriedades. Depois, na aba Sinais, associe o manipulador `ao_salvar` ao sinal de clique. Para ativar o último botão de forma a permitir que os usuários saiam do programa, selecione o valor da propriedade Botão de Estoque como Sair e depois associe o manipulador `ao_sair` a seu sinal `clicado`.

Seletor de arquivos

As barras de menu e de ferramentas já estão finalizadas, e a interface gráfica da nossa ferramenta de captura de tela está quase completa. O usuário precisa de uma forma de especificar o nome de arquivo a ser usado para salvar a imagem gerada. O programa poderia usar um diálogo convencional de gravação de arquivo, mas é mais ágil integrar essa funcionalidade diretamente na janela do aplicativo. Clique no widget Seletor de Arquivos na categoria Adicional da paleta. Em seguida, adicione-o à janela do aplicativo. O arquivo-alvo de nosso seletor de arquivos será acessado pelo método `ao_salvar`, então o widget precisará de um nome que possa ser lembrado. Especifique seu nome como `seletorDeArquivos`, e altere o valor da propriedade Ação para Save.

A propriedade Somente Local indica se o widget será capaz de acessar arquivos remotos. Para realizar essa tarefa, o aplicativo final terá que fazer uso do sistema de arquivos virtual do Gnome, o Gnome VFS, que é capaz de ler e gravar arquivos remotos com diversos protocolos, incluindo FTP, SSH e SMB. Especifique o valor de Somente Local do widget seletor de arquivos para Não, pois

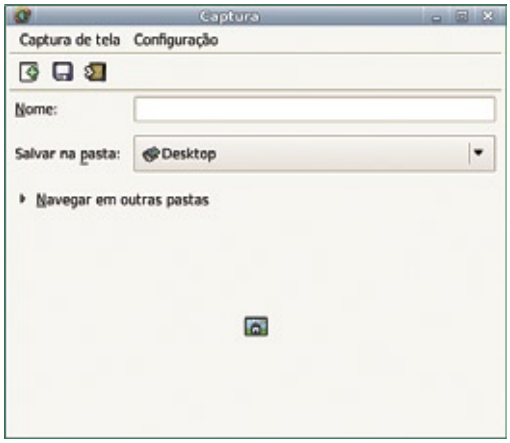

Figura 6 A interface finalizada do aplicativo de captura de tela está pronta para inclusão num programa em *Ruby*.

é desejável que o usuário consiga gravar suas capturas diretamente num servidor FTP para divulgá-las na Web.

Queremos incluir o recurso de visualização da captura em tela cheia, e não apenas como uma simples miniatura, pois esse é o maior defeito da ferramenta nativa do Gnome. Para isso, precisaremos de barras de rolagem.

Alguns widgets GTK já incluem suporte a barras de rolagem, mas outros exigem sua inclusão através de um widget de Porta de visualização.

Clique no ícone de Porta de visualização na categoria Básico, e depois no último ícone da última linha da Paleta. Em seguida, clique no último espaço vazio na parte de baixo da janela do futuro aplicativo. As barras de rolagem aparecerão nas bordas do espaço vazio. Por padrão, as barras sempre ficarão

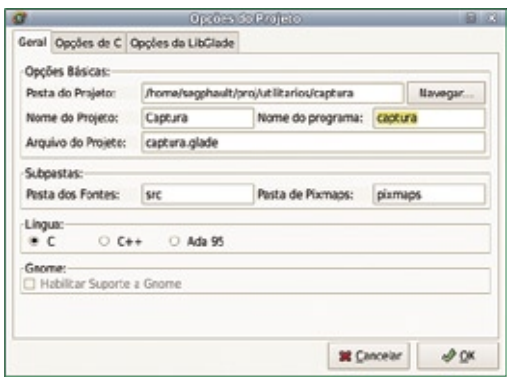

Figura 7 O diálogo de *Opções do Projeto* permite que os usuários selecionem um diretório e um arquivo de destino para os projetos do *Glade*.

Exemplo 1: captura.rb

```
01 #!/usr/bin/env ruby
02
03 # Carregar as bibliotecas necessárias
04 require “RMagick”
05 require “libglade2”
06 require “gnomevfs”
07
08 # Inicializar o toolkit GTK
09 Gtk.init
10 # Especificar a localização da captura temporária
11 $arquivo_de_captura = File.join(ENV[“HOME”], “.captura_de_tela.png”)
12 # Especificar a localização do arquivo de interface do Glade
13 $arquivo_glade = File.join(File.dirname(__FILE__), “captura.glade”)
14
15 def ao_sair
16 # Sair do programa
17 Gtk.main_quit
18 end
19
20 def ao_capturar
21 # Capturar a tela e salvar a imagem no disco em $arquivo_de_captura
22 Magick::Image.capture($glade [“checkBeep”].active?, $glade[“checkFrame”].
   ➥active?).write($arquivo_de_captura)
23 # Carregar a imagem no local de pré-visualização
24 $glade[“imagePreview”].file = $arquivo_de_captura
25 end
26
27 def ao_salvar
28 # Abrir o arquivo não-exclusivamente e em modo de escrita
29 f = GnomeVFS::File.new($glade[“fileChooser”].uri, 2, false)
30 # Gravar o conteúdo do arquivo temporário num destino
31 f.write(File.open($arquivo_de_captura).read)
32 # Fechar o arquivo de destino
33 f.close
34 end
35
36 # Carregar o arquivo do Glade
37 $glade = GladeXML.new($arquivo_glade) {|h| method(h)}
38
39 # Iniciar o loop principal
40 Gtk.main
```

visíveis. Para fazer com que somente apareçam quando forem necessárias, altere as políticas V e H para Automatic. Feito isso, é necessário adicionar o widget de imagem à porta de visualização, selecionando para isso o ícone de widget de imagem na categoria Básico. Agora, clique no espaço vazio que receberá o widget de imagem. O conteúdo desse widget será manipulado no código-fonte do programa, e portanto necessita de um nome com significado. Insira o valor `previewImagem` na propriedade Nome.

Para ter certeza de que o programa terminará quando o usuário fechar a janela, é necessário associar um manipulador ao evento de destruição da janela. Para isso, selecione o item *janelaCaptura* na árvore de widgets, navegue até a aba Sinais e selecione *destroy* no diálogo que se abre. Em seguida, digite `ao_sair` no Manipulador e clique no botão Adicionar. Com isso, o programa invocará o método `ao_sair` quando o usuário fechar a janela.

Salvando o projeto

Antes de usar a interface (**figura 6**) em algum script em Ruby, é necessário salvar o projeto do Glade. Para tal, selecione Salvar no menu Projeto da janela de projetos. No diálogo Opções do projeto (**figura 7**), selecione o diretório onde o projeto do Glade será salvo, e digite `Captura` no Nome do Projeto. Ignore as outras opções desse diálogo, pois referem-se a recursos obsoletos de geração de código. O arquivo `.glade` gerado contém os dados em XML relacionados à interface gráfica. Também será gerado um arquivo `.gladep`, que armazena outras informações do projeto.

Vamos programar?

Para transformar o arquivo de interface do Glade em um aplicativo Ruby plenamente funcional, será necessário criar um simples script em Ruby que carregue o arquivo do Glade, exiba a janela e associe vários eventos de interface a funcionalidades do aplicativo.

Comece criando um arquivo chamado `captura.rb` conforme o **exemplo 1**. Salve o arquivo no mesmo diretório que o arquivo `captura.glade`. O script carrega as bibliotecas necessárias com o termo `require`. No caso, usaremos a *RMagick* para capturar a tela, a *libglade2* para lidar com o arquivo de interface gerado pelo Glade e a *gnomevfs* para fornecer acesso transparente à rede. Depois de carregar a libglade2, o script precisa inicializar a infraestrutura GTK subjacente, o que é feito com a linha `Gtk.init`.

Quando o programa captura uma tela, ele precisa gravar o arquivo de imagem num local temporário. Por conveniência, vamos fazer isso no diretório home do usuário, com o nome `.captura_de_tela.png`. Após inicializar a GTK, a variável `capture_file` recebe o caminho do local de armazenamento temporário, através do método `File.join` e a variável de ambiente `HOME`. O método `File.join` combina arquivos e caminhos de diretório com o separador de diretórios normal, (/ em sistemas Linux). Um `ENV[“HOME”]` retorna o caminho completo do diretório home do usuário.

Atribuído o valor correto à variável `$arquivo_captura`, o programa precisa especificar a localização do arquivo de interface do Glade. O método `File.join` é usado novamente, mas dessa vez com

o método `File.dirname` e o nome do arquivo do Glade (`captura.glade`).

Métodos

Agora precisamos implementar os métodos `ao_sair`, `ao_capturar` e `ao_salvar`. Eles devem ser definidos antes de ser invocado o comando `GladeXML.new`, que associa esses métodos a sinais.

O método `ao_sair` contém apenas uma chamada de método, `Gtk.main_quit`, que fecha o programa. Se quiséssemos fazer o programa abrir um diálogo para perguntar se o usuário realmente deseja fechá-lo, seria este o ponto para fazê-lo.

O método `ao_capturar` é um pouco mais complexo. Sua primeira linha emprega o método `Magick::Image.capture` para capturar a tela, e depois o método `write` para salvar a imagem no arquivo temporário descrito na variável `$arquivo_de_captura`.

A segunda linha carrega a tela capturada no painel de pré-visualização. O método `Image.capture` do módulo Magick recebe dois parâmetros, que especificam se o programa deve apitar após bater a foto, e se deve ser incluída a moldura da janela selecionada.

Para determinar se as checkboxes do menu Configuração estão marcadas, é necessário acessar esses widgets a partir do objeto Glade armazenado na variável `$glade` e então usar o método `active?`.

São usados colchetes para acessar um widget de um objeto Glade. Por exemplo, para acessar a checkbox `checkFrame`, usaríamos `$glade ["checkFrame"]`. No método `ao_capturar`, também chamamos o método `active?` com o widget `checkFrame` depois de acessá-lo pelo objeto Glade, com `$glade ["checkFrame"].active?`. Faremos isso tanto para a configuração de `checkFrame` quanto para `checkBeep`, passando os valores retornados, booleanos, para o método `Image.capture` como parâmetros.

A última linha do método `ao_capturar` acessa o widget `imagePreview` e então associa sua propriedade de arquivo ao nome de arquivo guardado na variável `$arquivo_de_captura`.

O método `ao_salvar` salva a imagem capturada na localização especificada pelo usuário com o widget seletor de arquivo integrado. Ele também usa o sistema de arquivos virtual do Gnome para oferecer suporte ao acesso transparente a arquivos pela rede, que permite que os usuários salvem arquivos em localizações remotas com vários protocolos, incluindo FTP e SMB.

O método `GnomeVFS::File.new` abre um arquivo na localização especificada para leitura ou escrita. O primeiro parâmetro enviado ao método `GnomeVFS::File.new` é o caminho completo do arquivo a ser aberto, que é obtido a partir da propriedade URI do widget seletor de arquivos. O segundo parâmetro determina se o arquivo está aberto em modo de leitura ou de escrita. Nesse caso, usamos `2`, que indica o modo de escrita.

O terceiro parâmetro da chamada de `GnomeVFS::File.new` especifica se o programa tem permissão para sobrescrever arquivos pré-existentes. No **exemplo 1** é usado `false` como valor desse parâmetro, indicando que arquivos pré-existentes podem ser sobrescritos.

A segunda linha do método `ao_salvar` abre e lê o arquivo descrito na variável `$arquivo_de_captura` e o grava no novo arquivo criado pela chamada a `GnomeVFS::File.new`. A última linha do método fecha o novo arquivo.

Como usamos o GnomeVFS em vez dos métodos internos de Ruby para salvar o arquivo de imagem na localização determinada, os usuários podem salvar os arquivos em servidores remotos ou outros computadores da rede local. Para tirar proveito dessa funcionalidade, um usuário Gnome primeiro precisa conectar-se a uma máquina remota selecionando Conectar ao Servidor... no menu Locais do Gnome. Se ele montar uma localização remota com o diálogo do Gnome, a localização ficará acessível no componente seletor de arquivos em nosso utilitário.

Exemplo 2: Execução do programa

```
01 $ chmod +x captura.rb
02 $ ./captura.rb
```

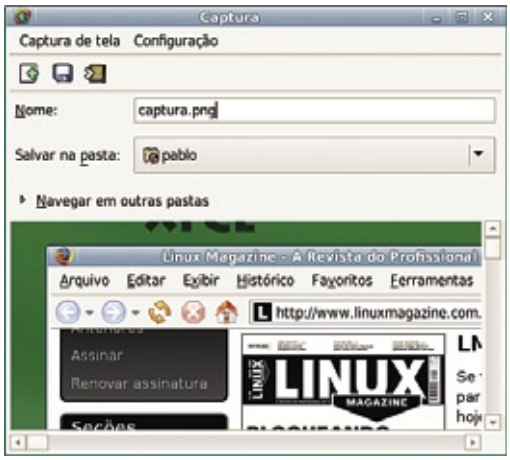

Figura 8 O aplicativo totalmente funcional está útil, apesar de sua simplicidade.

Executando

Agora que nosso aplicativo está finalizado, é hora de pô-lo à prova. Para executá-lo, mude as permissões do script Ruby para torná-lo executável, e depois rode-o como um script de shell normal (**exemplo 2**). O programa inteiro, com apenas aproximadamente 20 linhas de código, possui mais funcionalidades que o capturador de telas interno do Gnome (**figura 8**). Com Ruby e Glade, os desenvolvedores conseguem produzir aplicativos realmente úteis com mínimo esforço e grande eficiência. E o melhor de tudo é que o uso dessas duas ferramentas elimina a necessidade de compilação. O programa pode ser executado imediatamente após a modificação, descartando a necessidade de qualquer passo intermediário.

O Glade traz diversos widgets adicionais não discutidos neste artigo. Pode-se experimentar diversas ferramentas da paleta para aprender mais a respeito das funcionalidades oferecidas pelo fabuloso Glade. A documentação do Gnome relativa ao desenvolvimento de software também fornece importantes informações sobre widgets específicos. ■