MacPRO
o suplemento dos power users
Ano 2 - Nº19
Parte integrante da revista Macmania
Não pode ser vendido separadamente

ProNotas

Update do InDesign grátis

Quem comprou a versão 1.0 vai receber a nova na faixa

Primeiro, a Adobe anunciou uma atualização do **InDesign**, o 1.5 (para Mac OS 8.5, 8.6 e 9; Windows 98, 2000 e NT). Isso deixou os usuários felizes, pois a nova versão trazia correções para alguns bugs do programa. Mas a empresa afirmou que todo mundo teria que pagar US$ 99 pelo upgrade, pois o InDesign 1.5 não seria só uma correção de problemas, mas sim também um conjunto de novas funções. Daí o caldo engrossou. Queixas começaram a pipocar na Internet, com muitos usuários profundamente irados com a Adobe, exigindo que o novo software fosse de graça para quem já tinha adquirido a primeira versão.

Agora, ao que parece, a situação parece ter finalmente sido contornada: o presidente da Adobe, Chuck Geschke, disse que aqueles que compraram o InDesign 1.0 (e pagaram US$ 699) terão a nova versão de graça. A decisão foi tomada depois que o próprio Geschke leu as declarações raivosas dos consumidores em listas de discussões na Web.

Juntamente com o anúncio, a Adobe aproveitou para revelar que teve o melhor primeiro trimestre fiscal de sua história, com lucros de US$ 64 milhões – um aumento de 57% em relação ao mesmo período do ano passado.

As principais novidades do InDesign 1.5 são: maior integração com os outros produtos Adobe; novas ferramentas para melhorar a criatividade e a produtividade; e um controle melhor e mais preciso do produto final. Essa integração servirá como atrativo para a adoção do programa pelos profissionais, graças ao refinamento no fluxo de trabalho de quem já usa vários outros produtos profissionais da Adobe, como o PressReady (ferramenta de impressão) e o InProduction (pacote com cinco programas, que são uma extensão do Adobe Acrobat no que tange à conversão de cores e definição de parâmetros de trapping em arquivos PDF).

Outras inovações são: texto em um path, *trapping* embutido, impressão de PDF, ferramenta lápis, um gerenciador de plug-ins e uma ferramenta de *free transform*. Vários desenvolvedores já estão providenciando plug-ins para o InDesign para melhorar a funcionalidade da nova versão.

Além disso, a Adobe lançou a versão em japonês do InDesign durante a Macworld de Tóquio, e as versões alemã e internacional devem sair logo depois que o novo InDesign chegar às lojas norte-americanas e canadenses, o que deve acontecer em abril.

Adobe: www.adobe.com

Crie o seu plug-in de Sherlock

Parte 1 de 2

por Tiago Gimenez Ribeiro

Um dos recursos mais interessantes introduzidos no Mac OS 8.5 foi, sem dúvida, o Sherlock. Com ele, você pode realizar sofisticadas buscas por informações na Internet, através de uma interface amigável e descomplicada. Mas, a despeito do poder dessa ferramenta, muitos desenvolvedores brasileiros de conteúdo na Internet ainda não descobriram o quanto podem se beneficiar dessa tecnologia. Por essa razão, nesta série de duas matérias tentarei fornecer os subsídios necessários para que você entenda o funcionamento do Sherlock e possa desenvolver os seus próprios plug-ins de busca.

Como funciona o Sherlock?

Antes de sair por aí escrevendo plug-ins, é importante entender como o mecanismo funciona. Basicamente, a idéia consiste em passar para o aplicativo de busca (o Sherlock propriamente dito) os seguintes parâmetros:

1) O texto ou dado a ser encontrado.

2) As informações sobre o mecanismo de busca (*search engine*) que você irá usar para realizar a busca.

O primeiro parâmetro (o texto a ser encontrado) é de responsabilidade do usuário. Ou seja, ele deverá digitar no campo de texto apropriado qual é a informação que ele deseja procurar. Já o segundo tipo de informação (os dados da máquina de busca) irá, conforme você já deve ter adivinhado, variar conforme o serviço de busca pelo qual o usuário deseja fazer a pesquisa. Os dados de um determinado serviço de busca poderão mudar ao longo do tempo, exigindo a atualização constante dos dados.

Por essas razões, a Apple modularizou o Sherlock, de modo a tornar possível ampliar sua capacidade através de plug-ins de terceiros. Cada desenvolvedor pode escrever um arquivo de configuração de seu serviço de busca (o tal do plug-in), que é colocado na pasta de plug-ins do Sherlock (Internet Search Sites, localizada no System Folder) e

Informações sobre o Sherlock na Web

Página oficial do Sherlock:
www.apple.com/sherlock

Plug-ins de Sherlock:
www.apple.com/sherlock/plugins.html

Technote 1141 – "Extending and Controlling Sherlock":
http://developer.apple.com/technotes/tn/tn1141.html

Tutorial – "Writing a Sherlock Plugin", por Gord Lacey:
www.apple-donuts.com/sherlocksearch/howto.html

Lista de discussão sobre o Sherlock (não-oficial):
www.mdg.com/Sherlock/Sherlock-Talk.html

ProNotas

continuação

Macromedia apresenta UltraDev

Parecido com o Dreamweaver, ele foi concebido criar aplicativos na Web

Nem só de Flash vive a Macromedia. Na Internet World, a empresa mostrou para os desenvolvedores de páginas da Internet um novo programa que vai oferecer outra alternativa para construção de sites. O **UltraDev** está direcionado ao *e-commerce* (comércio eletrônico), uma das áreas na Rede que está em franca ascensão. Ele é uma nova tecnologia para criar aplicativos Web mais dinâmicos e atrativos (isto é, com animação).

Versão revista e melhorada do Drumbeat 2000, o software traz um visual muito parecido com o do Dreamweaver. Assim, quem já trabalha com ele não vai encontrar problemas na para se adaptar. Segundo os executivos da empresa, o UltraDev serve, por exemplo, para criar uma loja virtual com um visual mais movimentado e convidativo, usando o que há de mais moderno em produção de sites. O lançamento está previsto ainda para este semestre, tanto para Mac quanto PC.

A Macromedia também deixou claro, na sua apresentação na feira da Internet, que não deixará de dar suporte aos usuários dos computadores da Apple, principalmente no que diz respeito ao Mac OS X. De acordo com a empresa, antes do final do ano a grande maioria, senão todos os seus programas, já estarão sendo adaptados para o futuro sistema operacional.

Macromedia: www.macromedia.com

Suas fotos de família, na Web

Novo compactador de imagens comprime muito melhor que o JPEG

Durante a Internet World em Los Angeles, a LizardTech, uma empresa especializada em compactação de imagens para Windows, vai lançar um produto para Mac que vai permitir comprimir fotos com qualidade superior à do JPEG.

O **MrSID Photo Edition** pode transformar uma foto digital de 9 MB num simples arquivo de 200K (uma compressão de 40:1)! Criado para usuários profissionais que precisam mandar imagens gigantescas pela Internet, o MrSID agora vai ter uma versão de "consumo". Na verdade, são duas versões: uma grátis, com recursos limitados, e outra, comercial, que custa US$ 49.

Segundo John Grizz, presidente da empresa, o fluxo de imagens pela Web – algo em torno de 7 bilhões de arquivos digitalizados – deve crescer 50% nos próximos cinco anos. A lenda diz que a idéia surgiu quando a mãe de Grizz pediu um programa para mandar fotos do neto para toda a família, mas que não ficassem apenas com a resolução da tela. A partir desse simples pedido, a LizardTech teria começado a adaptar seu software profissional para o mercado amador. A versão para Mac terá um plug-in para o Photoshop.

Grizz afirmou também que está feliz de voltar a produzir programas para Mac. "A empresa era exclusivamente voltada para Macintosh quando surgiu, em 1990. Estamos ansiosos em voltar".

LizardTech: www.lizardtech.com

Crie o seu plug-in de Sherlock

continuação

permite o acesso do Sherlock aos serviços. E mais: o desenvolvedor pode especificar em seu plug-in um site no qual o Sherlock irá procurar por atualizações. Desse modo, se eventualmente a sua máquina de busca for modificada de um modo tal que o seu plug-in exija uma revisão, basta que você coloque a nova versão no endereço especificado e, automaticamente, o Sherlock irá baixar a atualização para as máquinas de seus clientes.

Por onde começar

Muito bem: digamos, então, que desejamos escrever um plug-in para o Sherlock. Por onde começar? Em que linguagem? Quais as ferramentas necessárias? Vamos responder a essas perguntas passo a passo...

Muitos desenvolvedores para a Web não sabem o que estão perdendo com o Sherlock

Parte 1: Obter ferramentas para a criação do plug-in

Antes de mais nada, precisamos de ferramentas para a codificação e criação de nossos plug-ins. Como iremos ver adiante, já existem hoje algumas ferramentas visuais que auxiliam na tarefa de criar plug-ins para o Sherlock. Mas, na maior parte dos casos, você somente precisará de um navegador Web (Netscape, Explorer etc.), de um editor de resources (ResEdit, Resorcerer etc.) e de um editor de textos (SimpleText, BBEdit, Tex-Edit etc.).

Com exceção do editor de resources, que você pode comprar (no caso do Resorcerer) ou baixar de graça no site de FTP da Apple (`ftp.apple.com/developer`), todas as outras ferramentas são gratuitas. Muitas vezes, elas já vêm pré-instaladas no seu Mac!

Parte 2: Obter informações sobre seu serviço de busca

Agora, precisamos de informações relativas ao serviço de busca.

Usando o seu navegador da Web predileto, vá até a página de busca desejada e salve, em formato HTML:

1) O código HTML da página que contém o formulário de busca.

2) O código HTML de uma página desse serviço de busca que apresente os resultados de uma busca qualquer (de preferência, com mais de um resultado).

Essas páginas contêm informações que iremos extrair para confecção do plug-in. É importante notar, principalmente em páginas com muitos frames, que você deverá sempre salvar o código (e o link) do frame no qual estiverem os campos de texto, menus e botões do formulário. Não salve os demais frames, pois eles não irão interessar.

Listagem 1

```
01  # © 1998 Apple Computer, Inc.
02
03  <search
04       name = "AltaVista"
05       action = "http://www.altavista.com/cgi-bin/query"
06       update="http://si.info.apple.com/updates/AltaVista.src.hqx"
07       updateCheckDays = 3
08       method = get>
09
10  <input name="pg" value="q">
11  <input name="kl" value="XX">
12  <input name="user" value="sherlock">
13  <input name="q" user>
14
15  <interpret
16       bannerStart="<!- BANNER START ->"
17       bannerEnd="<!- BANNER END ->"
18
19       resultListStart="<!- RESULT LIST START ->"
20       resultListEnd="<!- RESULT LIST END ->"
21
22       resultItemStart="<!- RESULT ITEM START ->"
23       resultItemEnd="<!- RESULT ITEM END ->"
24
25       relevanceStart="<!- RELEVANCE START ->"
26       relevanceEnd ="<!- RELEVANCE END ->"
27       >
28  </search>
```

Parte 3: Partir de um modelo

Agora que temos as informações necessárias, vamos usá-las para construir o nosso plug-in. Primeiramente, experimente abrir em um editor de texto um plug-in de Sherlock já escrito, para facilitar seu trabalho. A pasta de plug-ins do Sherlock no System Folder contém uma porção deles.

Na listagem 1 temos o código do plug-in do AltaVista. (Os números à esquerda não fazem parte do código, servem apenas como numeração a fim de facilitar a localização dos trechos.)

Como você pode ver, um plug-in de Sherlock é bastante parecido com um arquivo HTML. Ele é composto, essencialmente, por *tags* (<input>, <search> etc.), que irão definir as diferentes áreas de dados do nosso plug-in. Em um plug-in de busca do Sherlock, as tags devem fornecer

as informações necessárias para:
1) A execução das operações de busca no serviço em questão.
2) A interpretação dos resultados das buscas propriamente ditas.
As linhas de texto iniciadas pelo caractere # são ignoradas pela máquina de busca; normalmente, utilizamos esse caractere quando desejamos incluir um comentário em um determinado trecho de nosso código, ou quando desejamos desativar parte de nosso código durante a fase de testes do plug-in.
Por ora, basta salvar o documento que você abriu com um outro nome. Escolha algo que identifique o seu plug-in; por exemplo: "MeuPlugin.src".

Listagem 2

```
<FORM NAME=buscanaweb
ACTION=http://www.meusite.com/cgi-bin/
        buscanaweb METHOD=GET>
```

Parte 4: Entender como o plug-in é organizado

A tag `<search>`, assim como a tag HTML encontrada em arquivos .html, é o principal delimitador de seu plug-in de busca. Ela deve ser declarada no início do seu código (linhas 03 a 08), assim como no final (linha 28). Você deve fornecer para esta tag os seguintes parâmetros:

Listagem 3

```
10   <input name="pg" value="q">
11   <input name="kl" value="XX">
12   <input name="user" value="sherlock">
13   <input name="q" user>
```

• `name` (linha 04): Nome identificador do seu plug-in, que o Sherlock irá apresentar ao usuário quando solicitado. Você pode, desse modo, dar o nome que quiser para o arquivo do plug-in ("TiagoPlugBeta2v4.src" e assim por diante) e, simultaneamente, fornecer um segundo nome, mais explicativo, para ser apresentado para o usuário na lista de plug-ins ("Plug-in do Tiago", por exemplo).

O código do plug-in de Sherlock não é muito diferente do HTML

• `action` (linha 05): URL onde se encontra a máquina de busca que você irá utilizar em seu plug-in. No código em HTML do formulário que você baixou da Internet, essa informação deve estar dentro da tag `<form>` (veja o exemplo).
• `update` (linha 06): URL onde você irá disponibilizará as atualizações de seu plug-in. Esse parâmetro é opcional, mas recomendado.
• `updateCheckDays` (linha 07): Usado em conjunto com o parâmetro `update`, ele especifica com que frequência (em dias) o Sherlock deverá procurar por atualizações de seu plug-in.
• `method` (linha 08): Especifica qual é o método que o formulário usa. Da mesma forma que o parâmetro `action`, você encontrará esta informação na tag `<form>` do código HTML que você baixou da Web (veja um exemplo na listagem 2).
Uma vez que você tenha aberto o seu código com a tag `<search>` e tenha entrado todos os parâmetros relativos ao site de busca, você deve especificar para o seu plug-in os componentes do formulário (campos de texto, menus pop-up etc.) envolvidos no envio das informações para a máquina de busca, bem como os seus respectivos valores. No exemplo do plug-in do AltaVista, temos:
• `name`: Nome do componente do formulário especificado; deve ser igual ao nome existente no código HTML que você baixou.
• `value`: Valor atribuído ao componente em questão. Essa é a informação passada como parâmetro para o servidor realizar a busca na Internet.
• `user`: Esse parâmetro informa ao Sherlock que essa é a informação a ser passada para a máquina de busca.
Note, na listagem 3, que você pode ter mais de uma tag `<input>`; o número de tags irá variar conforme a quantidade de componentes existentes no formulário de busca. Alguns desses componentes podem ser invisíveis, mas, ainda assim, eles devem ser especificados em seu plug-in, pois poderão estar, por exemplo, fornecendo à sua máquina de busca parâmetros de funcionamento.

E agora?

Depois que o seu plug-in tiver realizado a busca, será necessário interpretar os seus resultados, separando cada resultado individual de modo a apresentá-lo em uma lista e processando informações. Essa é a parte mais trabalhosa do processo, pois não depende apenas de você, mas também da forma pela qual a máquina de busca organiza as informações.
As dicas sobre como proceder para interpretar os dados e – eventualmente – tornar a formatação da sua máquina de busca mais integrada com o Sherlock serão vistas na segunda e última parte desta matéria.
Até lá! **M**

TIAGO GIMENEZ RIBEIRO
tiago.r@apple.com.br
Trabalha no DRC da Apple Brasil.

O mistério da pasta que copiava

Curso de AppleScript, parte 11

por Maurício L. Sadicoff

Era uma tarde cinzenta e chuvosa. O telefone vermelho tocou no momento exato em que eu ensaboava minhas costas no chuveiro. Molhando a sala por completo, corri a atender. O problema era sério. Dona Mirtes gritava desesperada ao telefone, completamente estupefata.
– A pasta! A pasta copia! – gritava ela.
Imediatamente, tomei controle da situação. Pessoas naquele estado têm que ser tratadas com autoridade, ou não respondem.
Mandei que D. Mirtes procurasse uma cadeira e sentasse. Depois do providencial copo d'água com açúcar e das abanadas com seu leque de filó, D. Mirtes parecia mais calma e, entre soluços, desabafou:
– O Oscar, meu marido, saiu hoje de manhã como sempre faz, depois de checar o email. Fui trabalhar tranquilamente depois do café da manhã, preciso terminar de digitar a receita do bolo de laranja pra mandar pra Cotinha, o senhor sabe, a D. Cotinha do 301, pra ela fazer pra neta dela que vem este domingo visitá-la. Escrevi tudo diretinho e gravei no desktop como sempre, porque essas pastas são tão complicadas... Porém (D. Mirtes é uma das poucas pessoas que ainda usam "porém" no dia-a-dia), quando fechei o programa, não consegui encontrar o arquivo, porque estava por detrás de uma janela aberta com um nome esquisito, que começava com B. Tirei a pasta da frente para colocar meu querido arquivo na pasta de documentos, e arrastei o documento para lá, movendo-o. Qual não foi minha surpresa quando – cáspite! – um arquivo com o mesmo nome do meu apareceu na pasta de nome esquisito! Concluí que a pasta estava copiando sozinha, o que só pode significar que tenho assombrações no meu computador!
– O nome da janela esquisita era Backup? – perguntei, já desconfiando do problema.
– Sim!, exclamou D. Mirtes, levantando os braços de euforia e acertando em cheio, com o leque de filó, a cara de D. Cotinha, que entrava para ver que agitação toda era aquela.
Naquele momento se resolvia o mistério da pasta que copiava. Acalmei D. Mirtes e me congratulei em silêncio por ler a série de artigos sobre AppleScript na Macmania. A solução foi simples. Seu Oscar, o marido de D. Mirtes, notório macmaníaco, provavelmente lia a mesma coluna. Arregacei as mangas (tarefa difícil quando se está vestindo apenas uma toalha) e pus mãos à obra... Expliquei para D. Mirtes que Seu Oscar provavelmente tinha criado uma Folder Action para proteger os arquivos..."

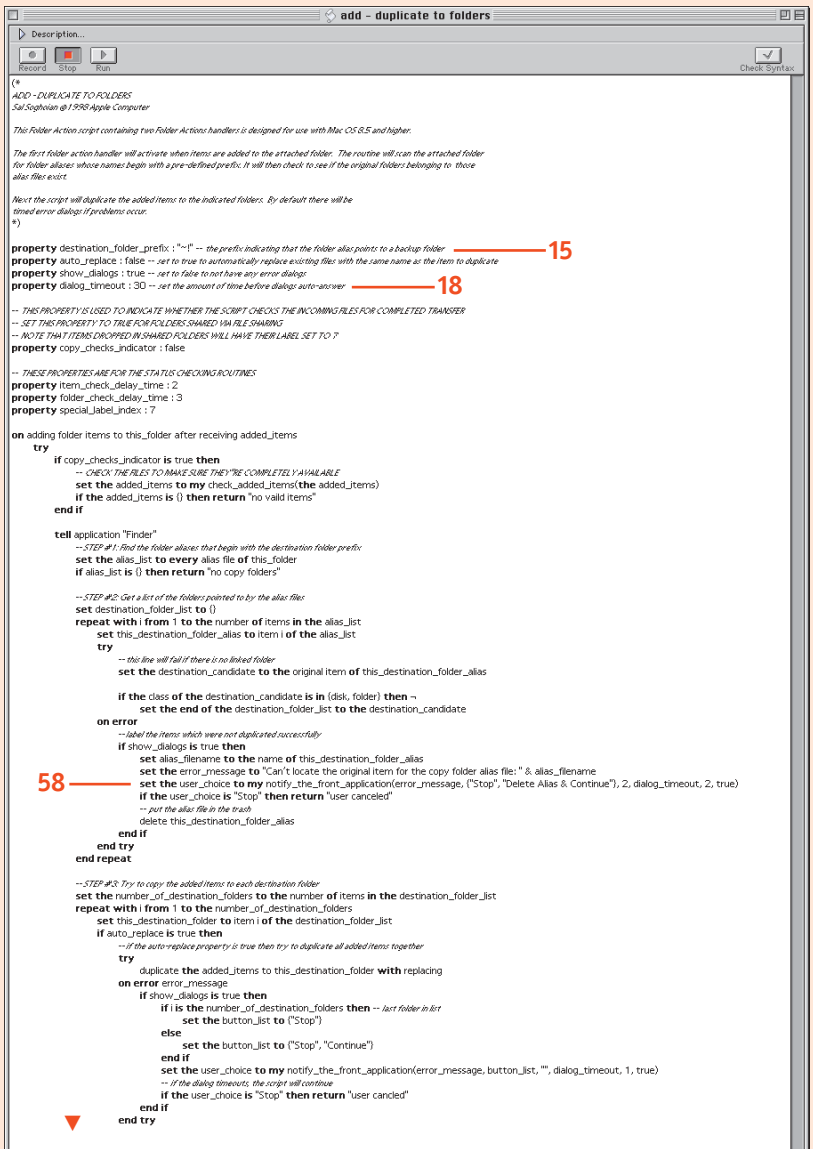
Trecho exclusivo para a MacPRO do livro em progresso **"Mistérios que Assombram Bits!"**, por Estêvão Trabalhos

Quantas vezes você não pensou que seria uma ótima ter uma maneira mágica de criar uma cópia de backup de um arquivo qualquer, no mesmo momento em que ele é criado? Todos que alguma vez tiveram a infelicidade de perder todos os dados numa falta de luz sabem do que estou falando.
Com as Folder Actions, usuários do Mac OS versão 8.5 (ou superior) têm essa função a seu dispor.
Uma das ações que vêm pré-programadas e guardadas na pasta Scripts dentro do System Folder tem o pomposo nome de add-duplicate to folders. No caso acima, Seu Oscar tinha associado a ação em questão à pasta Documentos. Colocou também, dentro da pasta, um alias para a pasta de Backup, com um til (~) na frente do nome. Como resultado, tudo que é jogado dentro da pasta Documentos é imediatamente copiado para a pasta de Backup, residente no desktop.
Essa pasta poderia estar residente em outro lugar (um disco na rede, um disco Zip ou até um CD-R), que o procedimento aconteceria da mesma forma. Assim que algo é colocado em Documentos, o sistema imediatamente o copia para a pasta Backup.

Esse script é o maior!

Se você vem acompanhando esta coluna assiduamente, compreender o script ao lado (que continua na página seguinte, por evidentes motivos de espaço) não será uma tarefa muito difícil (apesar do tamanho), graças à enorme quantidade de comentários previamente colocados pela equipe da Apple, que guiam a leitura do código. Portanto, vou apenas comentar as novidades desse script.
É bom notar também a quantidade de loops `try...else` para proteção e tratamento de erro, que raramente são tão detalhistas. É um excelente exemplo a ser seguido.

Linhas 15-18 – Aparece um verbo `property`, do qual nunca falamos. Está ali para definir propriedades, que podem ou ▶




```
(*
ADD - DUPLICATE TO FOLDERS
Sal Soghoian @1998 Apple Computer

This Folder Action script containing two Folder Actions handlers is designed for use with Mac OS 8.5 and higher.

The first folder action handler will activate when items are added to the attached folder.  The routine will scan the attached folder
for folder aliases whose names begin with a pre-defined prefix. It will then check to see if the original folders belonging to  those
alias files exist.

Next the script will duplicate the added items to the indicated folders. By default there will be
timed error dialogs if problems occur.
*)

property destination_folder_prefix : "~!" -- the prefix indicating that the folder alias points to a backup folder
property auto_replace : false -- set to true to automatically replace existing files with the same name as the item to duplicate
property show_dialogs : true -- set to false to not have any error dialogs
property dialog_timeout : 30 -- set the amount of time before dialogs auto-answer

-- THIS PROPERTY IS USED TO INDICATE WHETHER THE SCRIPT CHECKS THE INCOMING FILES FOR COMPLETED TRANSFER
-- SET THIS PROPERTY TO TRUE FOR FOLDERS SHARED VIA FILE SHARING
-- NOTE THAT ITEMS DROPPED IN SHARED FOLDERS WILL HAVE THEIR LABEL SET TO 7
property copy_checks_indicator : false

-- THESE PROPERTIES ARE FOR THE STATUS CHECKING ROUTINES
property item_check_delay_time : 2
property folder_check_delay_time : 3
property special_label_index : 7

on adding folder items to this_folder after receiving added_items
	try
		if copy_checks_indicator is true then
			-- CHECK THE FILES TO MAKE SURE THEY'RE COMPLETELY AVAILABLE
			set the added_items to my check_added_items(the added_items)
			if the added_items is {} then return "no valid items"
		end if

		tell application "Finder"
			--STEP #1: find the folder aliases that begin with the destination folder prefix
			set the alias_list to every alias file of this_folder
			if alias_list is {} then return "no copy folders"

			--STEP #2: Get a list of the folders pointed to by the alias files
			set destination_folder_list to {}
			repeat with i from 1 to the number of items in the alias_list
				set this_destination_folder_alias to item i of the alias_list
				try
					-- this line will fail if there is no linked folder
					set the destination_candidate to the original item of this_destination_folder_alias

					if the class of the destination_candidate is in {disk, folder} then ¬
						set the end of the destination_folder_list to the destination_candidate
				on error
					--label the items which were not duplicated successfully
					if show_dialogs is true then
						set alias_filename to the name of this_destination_folder_alias
						set the error_message to "Can't locate the original item for the copy folder alias file: " & alias_filename
						set the user_choice to my notify_the_front_application(error_message, {"Stop", "Delete Alias & Continue"}, 2, dialog_timeout, 2, true)
						if the user_choice is "Stop" then return "user canceled"
						-- put the alias file in the trash
						delete this_destination_folder_alias
					end if
				end try
			end repeat

			--STEP #3: Try to copy the added items to each destination folder
			set the number_of_destination_folders to the number of items in the destination_folder_list
			repeat with i from 1 to the number_of_destination_folders
				set this_destination_folder to item i of the destination_folder_list
				if auto_replace is true then
					--if the auto-replace property is true then try to duplicate all added items together
					try
						duplicate the added_items to this_destination_folder with replacing
					on error error_message
						if show_dialogs is true then
							if i is the number_of_destination_folders then -- last folder in list
								set the button_list to {"Stop"}
							else
								set the button_list to {"Stop", "Continue"}
							end if
							set the user_choice to my notify_the_front_application(error_message, button_list, "", dialog_timeout, 1, true)
							-- if the dialog timeouts, the script will continue
							if the user_choice is "Stop" then return "user cancled"
						end if
					end try
```

AppleScript

continuação

não ser usadas como variáveis. Propriedades, no entanto, podem ser relacionadas a objetos e pré-definidas no dicionário do aplicativo que criou o objeto – no caso, um alias. Esse é um conceito relativamente avançado em programação; portanto, basta saber que pastas, aliases, arquivos e que tais têm propriedades que podem ser definidas no começo de um script, como variáveis ou que são definidas previamente, quando da criação do objeto.

Linha 58 – Repare que estamos definindo o valor de uma variável, `user_choice`, baseado no valor da resposta que recebemos da função `notify_the_front_application`, que por sua vez é definida mais tarde no programa. Essa técnica é fundamental na organização de qualquer programa.
É sempre bom criar funções para fazer algo que será repetido várias vezes, ou para diminuir a confusão que acontece quando o código vira uma sequência ininterrupta de caracteres. Note que, para mostrar que a função era definida no próprio programa, usamos a keyword `my` antes do nome da função.

Linha 95 – Aqui há uma variação no uso do `try` – desta vez utilizado sem o `else`, mas com a opção `on error`. Aqui definimos um número de erro, a ser usado mais tarde para o caso de estarmos tentando copiar um arquivo já existente. Por que um número tão esquisito, aproximadamente -200000? Porque não queremos confundi-lo com os números de possíveis erros do sistema. Esse é um erro do script: serve para sairmos do `try` sabendo a causa do problema e possibilitando avisar-nos do que está acontecendo.

Linha 135 em diante – A partir daqui, várias funções são definidas para ajudar o programa principal, o qual analisamos acima. Essas funções são ferramentas que você provavelmente poderá usar em programas futuros. Vou comentar a única que não tem uma razão muito aparente.
Uma das minhas funções favoritas, `notify_the_front_application`, é uma função padrão em várias Folder Actions, porque quase todos os programas têm que, uma vez ou outra, mandar informação para o usuário, e às vezes esses programas estão rodando em *background* sem que o usuário se dê conta. Essa função existe para chamar a atenção do usuário quando há algum problema sério. Outra razão para ser uma função separada e não apenas um display dialog no meio do script é a facilidade de se copiar a rotina básica para vários scripts e depois modificar essa rotina básica adicionando código para resolver problemas específicos de um ou outro script.

Estude bem esse código que publicamos, se você quer ficar fera em AppleScript. Isso é código escrito pelos papas da linguagem, e às vezes é bem obscuro. Então, não se incomode se não pegar tudo da primeira vez. Só ter contato com esse nível de script já levanta a capacidade de qualquer um. No mês que vem vamos listar alguns scripts simples, mas úteis, para fazer coisas básicas como transformar um grupo de arquivos em *read-only*, por exemplo. **M**

MAURÍCIO L. SADICOFF
Cria websites na horta do fundo do seu quintal na Flórida, entre uma prova e outra do mestrado de Engenharia da Computação.

```
            else -- try duplicating the items one at a time to catch any errors
                set the number_of_added_items to the number of the added_items
                repeat with q from 1 to the number_of_added_items
                    set this_added_item to item q of the added_items
                    -- extract the item name from the file path
                    set this_item_path to this_added_item as text
                    set added_item_name to my extract_name_from_path(this_item_path)
                    try
                        -- check to see if file already exists; if so then error
95 ———————          if exists item added_item_name of this_destination_folder then error number -200152
                        -- duplicate the item
                        duplicate this_added_item to this_destination_folder
                    on error error_message number error_number
                        if show_dialogs is true then
                            if the error_number is -200152 then -- file already exists (-15267)
                                set the error_message to "The item: "" & added_item_name & ¬
                                    "" already exists in folder "" & the name of this_destination_folder & ""."
                                -- ask if the user wants to replace the existing item
                                set the user_choice to my notify_the_front_application(error_message, {"Stop", "Replace", "Skip"}, 3, dialog_timeout, 1, true)
                                if the user_choice is "Stop" then
                                    return "user canceled"
                                else if the user_choice is "" then -- > dialog timeout
                                    return "dialog timeout"
                                else if the user_choice is "Replace" then
                                    try
                                        duplicate this_added_item to this_destination_folder with replacing
                                    on error error_message
                                        set the user_choice to my notify_the_front_application(error_message, {"Stop", "Continue"}, 2, dialog_timeout, 1, true)
                                        if the user_choice is "Stop" then return "user canceled"
                                    end try
                                end if
                            else
                                set the user_choice to my notify_the_front_application(error_message, {"Stop", "Continue"}, 2, dialog_timeout, 1, true)
                                if the user_choice is "Stop" then return "user canceled"
                                if the user_choice is "" then return "dialog timeout"
                            end if
                        end if
                    end try
                end repeat
            end if
        end repeat
    end tell
on error the error_message
    set the user_choice to my notify_the_front_application(error_message, "", "", dialog_timeout, "", true)
    if the user_choice is "Cancel" then return "user canceled"
end try
end adding folder items to

-- STANDARD FOLDER ACTIONS VERSION
on notify_the_front_application(alert_message, button_list, default_button_name_or_index, dialog_timeout, icon_index, beep_indicator) ——— 135
    set the alert_message to "Folder Actions Alert: " & return & return & the alert_message
    if the button_list is "" then set the button_list to {"Cancel", "Stop"}
    if default_button_name_or_index is "" then set default_button_name_or_index to the number of items in the button_list
    --      tell application (path to frontmost application as text)
    if the beep_indicator is true then beep
    try
        if the icon_index is "" then
            display dialog alert_message buttons button_list default button default_button_name_or_index giving up after dialog_timeout
        else
            display dialog alert_message buttons button_list default button default_button_name_or_index with icon icon_index giving up after dialog_timeout
        end if
        set the user_choice to the button returned of the result
    on error
        -- a Cancel button was passed in the button list and chosen by the user
        set the user_choice to "Cancel"
    end try
    return the user_choice
    --      end tell
end notify_the_front_application

on extract_name_from_path(this_item_path)
    set this_item_path to this_item_path as text
    set AppleScript's text item delimiters to ":"
    set this_item_name to last item of the text items of this_item_path
    set AppleScript's text item delimiters to ""
    return this_item_name
end extract_name_from_path

on remove_labels(the added_items)
    tell application "Finder"
        repeat with this_item in the added_items
            set the label index of this_item to 0
        end repeat
    end tell
end remove_labels

on check_added_items(the added_items)
    -- check the transfer status of every added file to determine
    -- if each file has completed being moved into the attached folder
    set the notbusy_items to {}
    repeat with i from 1 to the number of items in the added_items
        set this_item to (item i of the added_items)
        if my check_busy_status(this_item) is false then
            set the end of the notbusy_items to this_item
        end if
    end repeat
    return the notbusy_items
end check_added_items

on check_busy_status(this_item)
    -- a folder can contain items partially transfered
    -- this routine will wait for all the folder contents to transfer
    if the last character of (this_item as text) is ":" then
        set the check_flag to false
        repeat
            -- look for any files within the folder that are still transferring
            tell application "Finder"
                try
                    set the busy_items to the name of every file of the entire contents of this_item ¬
                        whose file type begins with "bzy"
                on error
                    set the busy_items to {}
                end try
            end tell
            if the check_flag is true and the busy_items is {} then return false
            -- pause for the indicated time
            delay the folder_check_delay_time
            -- set the flag and check again
            set the check_flag to true
        end repeat
    else -- the passed item is a single file, suitcase, clipping, etc.
        -- check the label of the item. If it is the marked label, then it's already been processed so ignore
        tell application "Finder"
            if (the label index of this_item) as integer is the special_label_index then
                return "ignore"
            end if
        end tell
        set the check_flag to false
        repeat
            tell application "Finder"
                set the item_file_type to the file type of this_item
            end tell
            if the check_flag is true and ¬
                the item_file_type does not start with "bzy" then
                tell application "Finder"
                    set the label index of this_item to the special_label_index
                end tell
                -- allow the Finder time to change the label
                delay the item_check_delay_time
                return false
            else if the item_file_type does not start with "bzy" then
                -- set the flag and check again
                set the check_flag to true
            end if
            -- pause for the indicated time
            delay the item_check_delay_time
        end repeat
    end if
end check_busy_status


-- HANDLER FOR KEEPING THIS FOLDER OPEN
-- To activate the handler, remove the comment markers (the parens and asterisks) at the beginning and end of the following handler and save this script.
(*
on closing folder window for this_folder
    beep
    tell application "Finder"
        open this_folder
    end tell
end closing folder window for
*)
```