MAC OS X HACKS

100 Industrial-Strength Tips & Tools

O'REILLY®

Rael Dornfest & Kevin Hemenway

MAC OS X HACKS

MAC OS X HACKS

Rael Dornfest and Kevin Hemenway

O'REILLY®

Beijing · Cambridge · Farnham · Köln · Paris · Sebastopol · Taipei · Tokyo

HACK #57 Turning a Command-Line Script into an Application

What do you get when you combine the power of Unix scripting with the simplicity of the OS X GUI? A powerful droplet application limited only by your scripting prowess.

DropScript (*http://www.advogato.org/proj/DropScript/*), as the name suggests, is a little application onto which you can drop any shell, Perl, or other command-line script. It turns that script into a full-fledged, self-contained, double-clickable application capable of running on your desktop and doing interesting things with any files you feed it.

Perhaps an example is in order. I'll create a shell script to zip any files passed to it on the command line:

```
#!/bin/sh
gzip "$@"
```

I save it to *gzip.sh*, make it executable, and give it a whirl on the command line:

```
% chmod +x gzip.sh
% echo "something" > file1
% echo "something else" > file2
% ./gzip.sh file1 file2
% ls *.gz
file1.gz file2.gz
```

It works as expected, gzipping any files it's given.

Now I drag *gzip.sh* on to the DropScript application. Within seconds, a new application is created, called, suspiciously, Dropgzip (see Figure 5-23). This is a tiny application with all the functionality of my original *gzip.sh* shell script. Like its parent, it accepts files—only dropped onto it from the Finder rather than fed to it on the command line.

Figure 5-23. Creating a DropScript application, before and after

Yes, it's a simple example, but any script will work as long as it expects only files and folders as arguments.

Options

DropScript sports some simple options, embedded in the original script as comments. For example, while it makes sense that *gzip.sh* should accept any file or folder it's fed, *gunzip.sh* should accept only things that are zipped. To set this restriction in the script, you'd just add the following line:

```
# EXTENSIONS : "tgz" "tar" "gz" "Z" "zip"
```

Services

The most intriguing attribute of DropScript is that its applications can be made to export their functionality as services, appearing in the Services menu of just about any application.

To do so, specify a service name, like so:

```
# SERVICEMENU : "SomeService"
```

where *SomeService* is the name under which the service will be listed in the Services menu. You can even specify that a particular service live within a submenu by including a path in the option:

```
# SERVICEMENU : "SubMenu/SomeService"
```

Drop the script on DropScript and drag the resulting application to, where else but, your *Applications* folder. Log out and back in again and your new service will be right there in the Services menu, as shown in Figure 5-24.

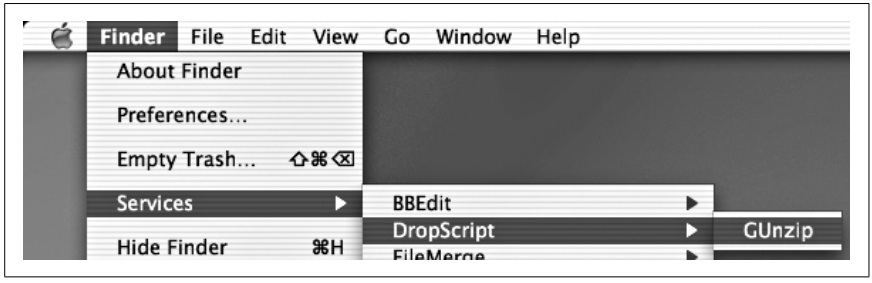

Figure 5-24. A DropScript application as exported service

I've only just scratched the surface of the sorts of applications you can build. After all, you have the power of all the built-in open source scripting languages (Perl, Python, Ruby, sh, etc.) at your disposal. You'll find some documentation and sample scripts (including a version of *gzip.sh*) in the *Examples* folder included with DropScript. These should be enough to get you started and experimenting.

See Also

- ScriptGUI (*http://homepage.mac.com/cnorris/ScriptGUI/about.html*), a similar Unix-script-to-GUI-application converter. It doesn't provide exported services, but does include a handy GUI window for running and inspecting scripts.