The journal of Apple technology.

Volume Number: 19 (2003)
Issue Number: 1
Column Tag: Mac OS X

CamelBones

Creating GUI-based apps with Perl

by Rich Morin

Sherm Pendley bills CamelBones as "An Objective-C/Perl bridge framework". The web page goes on to say "CamelBones is a framework that allows many types of Cocoa programs to be written entirely in Perl. It also provides a high-level object-oriented wrapper around an embedded Perl interpreter, so that Cocoa programs written in Objective-C can easily make use of code and libraries written in Perl."

Jumping from the general to the particular, CamelBones allows a Perl scripter to create GUI-based apps, with only a modicum of pain (read, Objective-C :-). Because the scripter can use Interface Builder, the graphic layout is painless, quick, and likely to yield attractive results.

Limitations

Sherm says that CamelBone's biggest current limitation is inheritance. It's not yet possible to create a Perl sub-class that inherits from an Objective-C sub-class. As a result, CamelBones can't be used for:

- Document-based applications, which require a subclass of NSDocument.
- Custom controls and/or cells, which require a subclass of one of the many NSControl or NSCell descendants, respectively.

This limitation will be addressed in the next version (0.3), which Sherm expects to have ready by the time this article is in print. Meanwhile, most of the tasks which could be performed by a cub-class can be handled by a Delegation or Notification callback.

CamelBones has some other problems, at least from my perspective. Because the app needs to be "built" (with Project Builder) rather than simply run, the build time becomes a noticeable part of the edit/test cycle. I think that's just the price we pay for using compilers and linkers (:-).

Also, most of the existing Cocoa documentation assumes that you're using Objective-C. This means that you have to have a reading knowledge of Objective-C, as well as the ability to turn method prototypes and example code into their Perl equivalents. I find this livable, however, and help is on the way for both problems

Getting Started

Assuming that you already have OSX and Apple's Developer Tools installed, adding CamelBones is quite simple. Go to http//camelbones.sf.net, download the distribution, and install it! At this writing, the documentation is still a bit sketchy, but the author plans to improve it Real Soon Now, so it may be in significantly better shape by the time you see it.

The package uses a standard installer, so installation is quite easy. I'd suggest that you choose the "Custom" install, so that you can get the source code (why not?). I haven't looked at the code, but I assume that it's a combination of Objective-C and Perl. Nice to have around, if you get frustrated by some peculiarity or simply become curious about how it all works.

If you are thinking about using CamelBones in a proprietary offering, you may want to look over the license a bit. CamelBones is released under the GNU Project's "Lesser General Public License" (LGPL). Briefly, the LGPL allows your app to use CamelBones without any requirement that the author provide the source code for the application (though with Perl, source distribution is the default case). On the other hand, any changes you make to CamelBones itself must be made available to anyone who gets the "binaries".

Once you have CamelBones installed, I strongly suggest that you skim the documentation and walk through the HowTo examples. As a newcomer to Interface Builder and Project Builder, this taught me a bit about the capabilities and operation of each. Even if you are expert at IB and PB, the examples will give you an idea of what kind of Perl code CamelBones expects.

In brief, however, Perl code for use with CamelBones looks pretty much like Perl code to work with any object-oriented API. Stuff like:

```
sub sayHello {
    my ($self, $sender) = @_;
    $self->{'TextLabel'}->setStringValue("Hello");
    NSLog("Hello, world!");
}
```

One peculiarity, discussed in the CamelBones web pages, is that CamelBones doesn't (yet) provide "toll-free" bridging of Perl hashes and lists to Cocoa's collection classes. Consequently, the Perl code has to deal with the Cocoa collections explicitly. Instead of writing:

```
my %hash;
my $key     = 'abc';
$hash{$key} = 'def';
printf("key=%s, value=%s\n",
  $key, $hash{$key});
```

you have to write something like:

```
my $hash = NSMutableDictionary->alloc->init;
my $key  = 'abc';
$hash->setObject_forKey($key, 'def');
printf("key=%s, value=%s\n",
  $key, $hash->objectForKey($key));
```

Also, because Objective-C doesn't provide automatic (Perl-style) garbage collection, you may need to explicitly reserve and release any Objective-C objects which you create. This looks pretty tedious, to my Perl-accustomed eyes, but I'm consoled by the facts that (a) I only have to use Cocoa collections to access Cocoa-specific items and (b) relief is said to be on the way.

Packaging Issues

Ease of distribution and installation has been a CamelBones priority from the start. All that the end user needs is a copy of CamelBones.framework. By default, applications look for this in /Library/Frameworks, so the easiest thing to do is to create an installer package that will make sure that the framework can be found there.

But, if a developer wants a drag-and-drop install and is willing to rebuild the framework with the appropriate Project Builder options, the framework can be embedded in the application bundle itself. CPAN (Comprehensive Perl Archive Network) modules can also be included in the application bundle, eliminating another common source of pain for end users of Perl programs.

Version Creep

The CamelBones framework is linked against the Perl interpreter (Version 5.6.0) that is shipped with OSX. Sherm has gotten reports that Perl 5.6.1 works fine, as long as it's compiled and installed identically to the original 5.6.0, but 5.8.0 definitely doesn't. So, if you have added Perl 5.8.0 to your system, you may have to rebuild any CamelBones apps you receive.

As time goes on, Apple is quite likely to release Perl 5.8.0 (or whatever) as an update to OSX. Unless a workaround is found, this will cause all 5.6.0-based CamelBones apps to break. Fortunately, some Very Bright People are looking into the matter, so a fix is likely.

Resources

Perl wizard Dan Sugalski is currently writing "Programming Cocoa Applications with Perl" for O'Reilly, but a publication date has not yet been established. The book will cover CamelBones in depth, however, so watch for it! In the meanwhile, Dan promises to put up some example code on the CamelBones web site.

Notwithstanding tha fact that they assume Objective-C usage, most books on Cocoa, IB, and PB are relevant to CamelBones Here are some you might want to look over:

"Building Cocoa Applications: A Step-by-Step Guide"

Garfinkel and Mahoney

O'Reilly and Associates, 2002

ISBN 0-596-00235-1

"Cocoa Cookbook for Mac OS X"

Bill Cheeseman

Peachpit, 2002

ISBN 0-201-87801-1

"Cocoa Programming for Mac OS X"

Aaron Hillegass

Addison-Wesley, 2001

ISBN 0-201-72683-1

"Learning Cocoa with Objective-C", second edition

James Duncan Davidson

O'Reilly and Associates, 2002

ISBN 0-596-00301-3

There is also a 200+ page rundown on Objective-C, right on your Mac OS X system (/Developer/Documentation/Cocoa/ObjectiveC/ObjC.pdf). You should also consider joining the MacOSX-Perl email list, which covers CamelBones and other Perlish topics on Mac OS X. Send email to macosx-subscribe@perl.org to get started.

Although Apple provides documentation on the AppKit and Foundation frameworks, it can be tedious to navigate. So, pick up a copy of Hoshi Takanori's Cocoa Browser app (http://homepage2.nifty.com/hoshi-takanori/cocoa-browser). This will let you browse the Objective-C method descriptions in a speedy, hierarchically-based manner.

As noted above, you will still have to convert the method synopses into Perl format, but that's life. Actually, I'm actually working on a conversion app, but you'll have to wait until next month to read about it. Until then, happy hacking...

Rich Morin has been using computers since 1970, Unix since 1983, and Mac-based Unix since 1986 (when he helped Apple create A/UX 1.0). When he isn't writing this column, Rich runs Prime Time Freeware (www.ptf.com), a publisher of books and CD-ROMs for the Free and Open Source software community. Feel free to write to Rich at rdm@ptf.com.