Mavrig

a Tcl application construction kit

Jean-Claude Wippler
Equi 4 Software, NL

EuroTcl 2008, Strasbourg, FR

Let's write an app

- Tons of packages to build with - Tcllib, etc
- Choose:
  - file structure, dev vs release dirs
  - packages & extension versions
  - naming conventions
  - inter-subsystem connections
- and an object system, and a database, and ...

So many details

- Haven't written one line of code yet!
- Many - boring - choices
- Can't easily revisit these choices later
- Can I at least re-use my next code?
  - Nope, because I made lots of choices...
- Give me a way out, please!

DNTO

- I don't need another "framework"
- I want to retain full control of my app
- I just want more convenience out of the box
- But above all… Do Not Take Over

- Utility code, conventions, "evolvability"

What's an Application?

- Command Line
- GUI
- Network Server
- Network Client
- Web-App 2.0!
- Embedded
- My favorite procs
- New Packages
- C/C++ Extensions
- Plug-ins, "Themes"
- CONVENTIONS...

Does it matter?

- In theory, no - in practice, you bet
- So many silly choices up front
- Choices hamper change and "agility"
- Choices are the enemy of scripting
- Why can't we just write code & tests?
- Yet allow for changes as you gain insight

Starkits

- Place all your stuff in a "vfs" directory
- Add "main.tcl" & put extensions in "lib/"
- Develop as usual, nothing changes
- When ready - wrap and ship - period!
- But we could do so much more...

What can we learn?

- Some conventions are real time savers
- Use them and you get tools to help - SDX
- New conventions can benefit everyone
  - Starpacks - exe's - were added later on
- So one trick seems to be...

Pick a few conventions

- Convention → Second Nature
- Second Nature → More Time To Think !
- But… DNTO

- I want conventions to fit in nicely
- If natural, they'll move down my spine

Rig

- It's 1 Tcl script, defining 1 command
- A couple of - hopefully - useful procs
- Some - hopefully - useful conventions
- A bit of - hopefully - useful machinery

- DNTO - Rig's purpose is to serve and help

Naming can kill you

- If it's re-usable, it has to be named right
- Rig.tcl:
  - file name == module name == namespace
  - "Rig cmd ..." - no global var pollution
  - a few other cmds, if not defined by you
- Rig is a "module" and supports lots of 'em

Modules, i.e. "rigs"

- Rig will load file "Cool.tcl" as a module:

```
namespace eval Cool {
   namespace export -clear {[a-z]*}
   namespace ensemble create
   source /path/to/Cool.tcl
}
```

- Uses ensembles from Tcl 8.5:

  "Cool::abc 1 2 3" → "Cool abc 1 2 3"

- To auto-load, call "Rig modules /path/to" once

Convenience

- If MyModule.tcl contains this:

```
proc shout {msg} {
   puts [string toupper $msg]
}
```

- Then I can use it anywhere in my app as:

  MyModule shout "Have a nice day!"

- Each module acts as a singleton object
- Lower-case procs are its "public methods"

Simplicity

- Don't add more machinery than needed
- If you can't remember it, then forget it!

- Rig.tcl is a single file:
  - App: your stuff + "Rig.tcl" (+ "main.tcl")
  - All Rig "features" are in the "Rig" cmd
  - Auto-loading & auto-downloading

main.tcl

- "main.tcl" defines your policies for Rig
- Here's a simple but complete version:

```
source ./Rig.tcl
Rig modules ./rigcache ?http://<URL…>/?

Rig event main.Init $argv
Rig event main.Run
if {![info exists Rig::exit]} {
  vwait Rig::exit
}
Rig event main.Done $Rig::exit
exit $Rig::exit
```

Command Line

- hello1.tcl

```
puts "Hello, world!"
exit
```

- What happened?
  - Rig auto-loads all *.tcl files next to it
  - Rig defined a module called "hello1"
- But that's cheating - hello1.tcl is stupid

Rig-aware cmd-line

- hello2.tcl:

```
Rig hook main.Run {
   puts "Hello, world!"
   set ::Rig::exit 0
}
```

- What happened after loading?
  - "main.tcl" triggers "main.Run" event
  - When hook is called, do a clean exit

Why bother?

- Same "main.tcl" can also be used for:
  - Tk app, net server, net client, web app
  - Or any mix of them...
  - E.g. a socket for TkCon remote access
  - visit website for several examples
- One less app choice to make up front

Web Apps - Part 1

- A web server in 6 lines of Tcl:

```
Rig hook main.Run {
   Httpd start 8080 [namespace code Req]
}
proc Req {obj} {
   $obj respond "<h1>Hello, world!</h1>"
}
```

- What happened?
  - Rig auto-downloaded the Httpd rig
  - That module implements web server core

Web Apps - Part 2

- A few modules, each only 100's of LOC:
  - Httpd: HTTP server, each req is an obj
  - Render: Mason-like template engine
  - Wikify: Text to HTML converter
- Minimal dependencies, you glue it together
- Any others? You bet. Yours. Be creative!

Turbo development

- Bonus - a major productivity boost:
  - No more edit-run-debug cycles
  - Keep editor + browser + app running
  - Turbo development: edit-run-edit-run
  - … will give a demo in the break …
- It's all based on auto-reloading modules

Mavrig

- If Rig is a step in the right direction…
- …then Mavrig wants to take this further

- Mavrig = Modules And Views with Rig
  - Good modularity leads to code re-use
  - Views are about rich data structures

Modularity

- API - all calls are "module cmd ..."
- Coupling - Rig events are used as glue
- Naming - make sure things don't clash

- Modularity takes effort up front
- Benefits later: test isolation, re-use, tools

Rig Collection

- I'm setting up a module / rig collection
  - Public, even if nobody else wants it
  - Contributors can get Subversion access
  - Create private rigs, check-in to re-use
- A rig is not just to encapsulate new code
  - Package wrappers, downloaders, builders

Why?

- I'm tired of seeing lots of great code "snippets" which aren't easily re-used
- All we need are a few conventions ...
- ... and a simple website to share modules
- Find good conventions for docs & tests
- Rig can be a catalyst - but even if no one else wants it, Rig scratches my own itch.

How?

- All it takes to auto-load your modules is:

  "Rig modules ./rigcache"

- Add an extra arg and it can auto-download:

  "Rig ... http://contrib.mavrig.org/"

- Downloaded only on first use, then it's local
- Module inter-dependencies work fine

When?

- Rig, docs, and a few modules on-line now
- Am calling this "Rig 0.x" so far...
- Currently using Rig for Mavrig myself
- Various Mavrig modules "in progress"
- Play with it, pull on it, find out the limits
- If it sucks, let me know how / what to fix

Thank you

- Website for (and built with) Rig and Mavrig:

  http://mavrig.org/

  (high-availability server in Nürnberg)

- Let's make it convenient to re-use Tcl stuff

  jcw@equi4.com