A Wibbly Wobbly Web Server

Colin Macleod – https://cmacleod.me.uk
CGM on Tcl wiki and chat

Why?

- My web service Newsgrouper (presented last year) was using the **tclhttpd** web server.
- In theory tclhttpd can use threads to handle multiple requests concurrently.
- But later I discovered this was not actually working, only one request was being processed at a time.
- This matters because some Newsgrouper requests can be processed in a millisecond, while other can take several whole seconds. We don’t want the fast requests to be held up by the slow ones.

Why? - more

- Tclhttpd is large, 15k lines of code, and has many facilitities I was not using.
- The size makes it difficult to debug problems like the thread issues I was having.
- So I started looking for a smaller, simpler substitute.
- I was already bypassing the whole domain system in tclhttpd and having my Tcl code handle everything.
- I just needed basic HTTP serving with a clean interface to Tcl, but including file uploading, and request logging.

What?

- I looked at various alternatives – sorry, Naviserver seemed far too big and complex.
- Andy Goth's "Wibble" looked the most suitable, but…
- The **zonehandler** API was not a good fit for my existing Newsgrouper code.
- It lacked request logging.
- It had not been updated since 2014.
- So I forked it – the result is not Wibble but is **Wibbly**!

Wibble to Wibbly changes

- Stripped out zonehandlers, replaced with an API closer to tclhttpd, based on Web Objects.
- Stripped out Wibble's "inter-coroutine communication system", replaced with coroutine::util from Tcllib.
- Stripped out most of Wibble's header encoding system, not needed.
- Added request logging, in the same format as tclhttpd.
- Added SCGI (Simple Common Gateway Interface) client code.

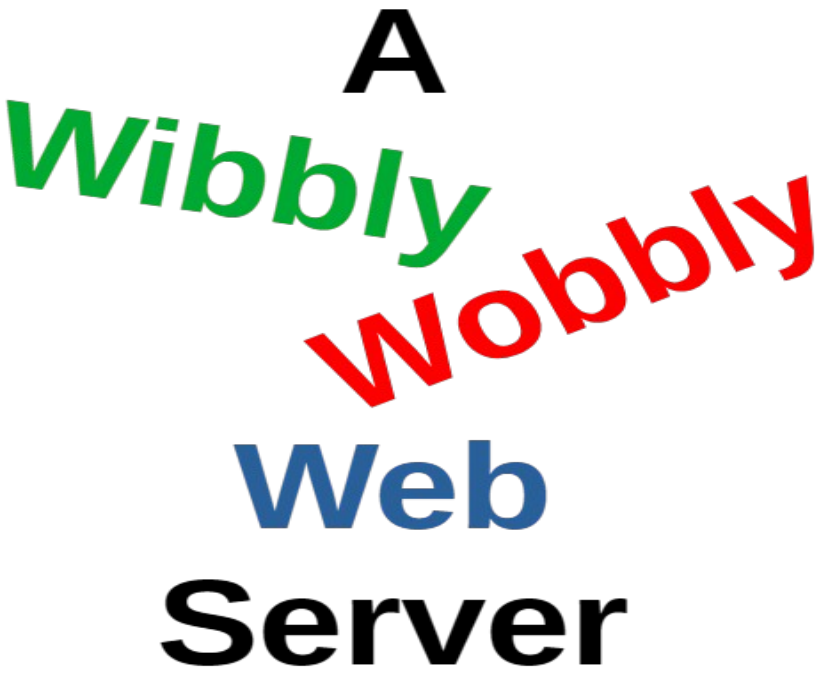
Wobbly API

- Application Tcl code handling a web request needs an interface to the web server.
- Wibbly provides this by passing a Web Object (wob) to the handler code. These **wob**s are of course members of the class **wobbly**.
- Wobbly has methods to find the path requested within the site being served, any cookies passed, form data sent, etc.
- Wobbly also has methods to set the response to return, to set cookies, or to return an error or redirect.

Wibbly Examples

- demo1 - default server.
- demo2 - serve static site.
- demo3a - generate survey form.
- demo3b - generate survey form, process responses.
- demo4a - serve fossil via SCGI.
- demo4b - serve fossil via SCGI, reject requests for php files.

Code can be found at https://cmacleod.me.uk/tcl/conf2026/ .



More info at: https://wiki.tcl-lang.org/page/Wibbly