Hoffmann RD

http://www.h-rd.org/

TWD – a simple TCL web dispatcher

Dr.ir. M. Hoffmann

Hoffmann RD
Wageningen
email: tcl@h-rd.org
© Hoffmann RD

talk EuroTCL 2009
TCL and the Web

- TCL has a lot of deployment options for server based Web applications.
- It seems that the use of TCL on the Web is declining.
- Most Web things are string based:
  - Generating html.
  - Reading and parsing request data.

Good match for TCL

- TCL has quite a few good database interfaces.
- AOLserver, Rivet, and mod_TCL are “complicated” to deploy on a standard shared hosting account.
- NEW: Woof! (I don’t know enough about it, but it seems to have a similar deployment scope.)
Background for TWD

- Develop a simple application for programmers.
- No server required (i.e. no own web server).
- Supply user and session data to programmer.
- Leave application development to programmer.
- A lot of potential to reuse existing TCL Web applications:
  - T's wiki
  - Rivet things
  - Wikit
  - THP
  - EFX
  - UCOME
Why TWD

Look at php et al:

- PmWiki: very nice application, but programming is counter-intuitive to plain text based pages.
- Drupal: It can do anything, but it is often easier to program something than to understand Drupal.
- OpenACS (TCL): Complicated to setup (except Debian) and understand.
- Wikit (TCL): Nice and simple, but on its own not enough for public web pages.
GOAL

Provide a

• simple foundation
• for programmers to build
• CUSTOMISED Web applications.
What is TWD

- TWD is based on T’s Wiki, an adaption of TiddlyWiki.
- TWD supplies a central place to dispatch to TCL proc, based on URL.
- SQLite db is used to store user data and session data:
  - SQLite handles concurrency.
  - SQLite: ACID.
  - SQLite is very good integrated with TCL and matches the string based paradigm of TCL.
- Currently TWD uses (N)CGI, planned are FCGI and SCGI.
- Starkit enabled.
What TWD is not

• AOLserver, use it when you need high performance AND you are willing to run your own server.
• mod_TCL, similar to above.
• NCGI, just supplies primitives for request handling etc., no user and session handling.
Why “no server required”? 

- Running a httpd (server) is a headache:
  - Is it up or down?
  - Does it leak memory?
  - Not possible on standard shared hosting, requires running your own (v)server -> even more work (security).

- Running on a DBMS (MySQL) means:
  - Deployment is more than simple file copy.
  - Changing hosting provider is more work.
  - Testing requires setup of server environment.
Potential uses for TWD

• Running TCL apps behind dispatcher allows e.g. authenticated Wikit.
• Embed calls to TWD in PmWiki.
• Integrate with email (SMTP, POP3, IMAP4).
• Simple database driven sites (mini OpenACS).
TWD invocation

- index.cgi: set up environment
- main.tcl: load required files and extensions
- twd.tcl: dispatcher
Dispatch

# process request
proc ::twd::main {} {
    # check user session
    session_check
    set path [::twd::getenv PATH_INFO ""]
    switch -glob -- $path {
        {} { ::tswiki::serve_wiki $action }
        /templates/* { serve_template $path $action }
        / { ::tswiki::serve_wiki $action }
        /* { serve_file $path }
    }
};# */
}
User db

-- user table
CREATE TABLE users(
    username TEXT PRIMARY KEY, -- username
    password TEXT, -- md5 password
    permissions TEXT -- user permissions
);

Very basic setup, can be extended by additional, programmer supplied tables.
Session db

currently cookie based, URL rewriting planned

-- sessions cookies
CREATE TABLE twdcookies(
  cookie TEXT PRIMARY KEY,  -- The login cookie
  username TEXT,            -- The user to log in as
  expires NUMBER,           -- When this cookie expires
  ipaddr TEXT,              -- IP address of browser
  agent TEXT                -- User agent of browser
);

Examples – simple template

<html><head><title>Tiny TWD time server</title></head>
<body><h1>Time server</h1>
Time now is: <%= [clock format [clock seconds]] %><br>
<hr>
</body></html>
Examples – check user permissions

# Process normal request or login/logout operation.
proc ::tswiki::tswiki_action_default {} {
    variable ::twd::u_permissions
    variable ::twd::body
    variable dir_tswiki_html
    if {!$u_permissions(read)} {
        # login page (no anonymous access)
        set body [subst -novariables \
            [::twd::read_template login.html $dir_tswiki_html]]
    } else {
        # normal wiki page
        set body [subst -novariables \
            [::twd::read_template wiki.html $dir_tswiki_html]]
        #log "$body"]}
Examples – T’s wiki actions

proc serve_wiki {action} {
    variable db $::twd::db
    switch -exact -- $action {
        {}                tswiki_action_default
        login             action_login
        logout            action_logout
        changepassword    action_change_password
        getuserlist       action_get_user_list
        updateuser        action_update_user
        gethistory        action_get_history
        save              action_save
        delete            action_delete
        rss               action_rss
        default           action_error}
}
Future Development

- Really integrate with Kit’s.
- Set up example site.
- Add session handling based on URL-rewrite.
- Settle down for ONE default TWD template mechanism.
- Increase coverage of test suite.
- Integrate VFS with SQLite: Web pages stored in VFS file (and at the same time in db).
- Set up source repository (Fossil or Berlios?).
Questions – Discussion – Suggestions

TWD is currently driven by my needs. Suggestions and ideas welcome. Open questions:

- Any drawbacks to require sqlite3?
- How are path’s handled in Kit’s vs. tclsh?
- A simple parser would be nice. Which?
- **SUGGESTIONS?**

TWD is (in part) based on T’s wiki and NCGI. Thanks.