AllegroServe and WebActions

An Idiosyncratic and Unsystematic Introduction

Robert P. Goldman

TC Lispers

2010-04-26 Mon

Introduction

- For lack of a better candidate, you get me talking about Franz's AllegroServe and Web actions.
- AllegroServe is a full web server written in Common Lisp.
  - AllegroServe is supposedly usable from other Common Lisp implementations. I have not tried this.
- Web actions is a dynamic web site bag of tricks.
- I am not a reliable guide!
  - I have not tried other frameworks.
  - I don't build real web applications (yet, at least).
  - I am very comfortable using ACL and like the environment a lot.

AllegroServe

- Multi-threaded web server (using Franz's concurrency features).
- Provides facilities for on-the-fly creation of web content using CL macros for HTML generation.
- Fairly basic set of facilities for doing richer web applications (but see web actions later on).
- Seems to me hugely preferable to wrestling with Apache!
- Caution: probably could get yourself in a lot of trouble, as far as security is concerned (but that's always true with the web).

Setting up a simple web-site

```
(require :aserve)
(net.aserve:publish-file
  :file "home:personal-web-page;index.html"
  :path "/rpg/index.html")
  => #<NET.ASERVE::FILE-ENTITY @ #x1000e45ba2>
(net.aserve:start :port 8000) =>
  #<NET.ASERVE:WSERVER port 8000 @ #x10010cb972>
```

Setting up a simple web-site

```
(require :aserve)
(net.aserve:publish-file
  :file "home:personal-web-page;index.html"
  :path "/rpg/index.html")
  => #<NET.ASERVE::FILE-ENTITY @ #x1000e45ba2>
(net.aserve:start :port 8000) =>
  #<NET.ASERVE:WSERVER port 8000 @ #x10010cb972>
```

...shows me without my silly picture...

```
(net.aserve:publish-file
  :file "home:personal-web-page;rpg.jpg"
  :path "/rpg/rpg.jpg")
```

...picture too...

Way easier

```
(net.aserve:publish-directory
  :prefix "/website"
  :destination "/Users/rpg/personal-web-page/")
;; logical pathnames don’t work :-(
```

- Supports access controls (specified in s-expressions!), HTTPS, cgi-bin, and other things we expect from a web server.
- XML-RPC server and client libraries.

Computed pages with s-expressions for HTML

```
(defpackage webex
 (:use common-lisp net.aserve net.html.generator))
(in-package :webex)
(publish
  :path "/rpg/page.html"
  :content-type "text/html"
  :function #'(lambda (req ent)
                (with-http-response (req ent)
                  (with-http-body (req ent)
                    (html
                      (:p "Hello World!"))))))
```

`html` macro lets us write HTML in lisp syntax.

Web applications

- AllegroServe provides (relatively) convenient query argument handling.
- `request-query` and `request-query-value` accessors for HTTP request objects.
- With this, build web pages that submit forms and other inputs for the in response to which the server computes new values in pages.
- Easy to build session objects on top of these accessors, for longer-term web applications.
- By generating pages with XML in them instead of HTML, can build AJAX applications that have richer interactions, don't block like conventional pages.

Advantages of AllegroServe

- You don't have to learn how to use Apache.
- Easy hot patch of running application.
- Excellent debugging of running web requests through emacs lisp interface.
- Turn on the debugger, interact with it while probing the web server with input that triggers an error, trap and find the error, fix it, recompile a bit of the lisp and keep going. No down time!

Rationale for web actions
(conjectured by me)

- The building blocks above are too rudimentary for easy construction of a web application.
- I have done this a couple of times and there's too much boilerplate of sessions to be managed, it doesn't capture the flow of the application, etc.
- Franz's web actions address these limitations.

Web actions features

- Web page templates;
- Session tracking;
- Application flow.

Web page templates

- AllegroServe can serve up normal HTML pages, or lisp-generated pages that contain HTML.
- Not convenient for pages that are *mostly* HTML and best generated by HTML tools, but with bits that are programmatically generated.
- Web actions adds Common Lisp Server pages. HTML pages that can have bits of CL included that are interpreted on the server side.
  - Define *clp functions* that are interpreted and interpolated before a page is served.
  - Like PHP, only it's lisp.

Session tracking

- Solves a problem left by AllegroServe:
  - Need to write code to track a session, typically by passing a magic number back and forth.
  - Need to associate information with a session.
  - Need to do this task without introducing errors or security holes.
- Can cause a web actions project to do session tracking.
- Websession objects are associated with requests, and you can attach *websession-variables* to them to add state to a session.

Application flow

- AllegroServe alone doesn't capture the high-level structure of a web application.
  - Typically move through a web application from page to page in a regular way.
  - Would like to see this captured and made visible.
  - Instead, each page's code has to have gotos in it.
- Web actions projects provide a high-level map that makes the structure of an application visible:
  - Simple maps associate symbolic names with particular pages. Action functions can direct the session accordingly.
  - Extended maps can have simple conditional logic — e.g., to do this action you must undergo a login check, then can proceed.

Things I'd like to see

- Why aren't session variables slots? Why don't they have accessors?
  - I have half-baked implementation of this in my own macros.
- More help with the client side.
- Provide login management integrated into web actions. Easy stuff to get wrong.
- Fuse database management into webactions.
- Not always obvious where session tracking will fall off.

Conclusions

- Quick skim over the features of Franz's web server libraries.
- Core AllegroServe available on other lisps — open-sourced; web actions seems proprietary.
- Based on my experiences, web actions provides a number of useful improvements over AllegroServe alone.
- AllegroServe, taken together with ACL's excellent debugging facilities, has worked out well for me.