

Smarmkets, Erlang, and web apps

Hunter Morris

hunter.morris@smarmkets.com

London Erlang User Group

21 May 2008

Skills Matter, London, UK

Erlang at Smarkets

- Betting Exchange
 - Scalable HTTP servers
 - High performance transactions
 - Many, many processes
 - API for third-party integration
 - “Real-time” event analysis

Existing Web Frameworks

- Yaws-based
 - ETCs Web Platform
 - Erlyweb
 - SPEWF
 - Jaws
 - ... vanilla appmods
- Mochiweb?
- Inets?
- Other servers?

Existing issues

- Reusability not encouraged
- Applications tied to servers
- Variation on the “wrong layer”

Introducing EWGI

- <http://code.google.com/p/ewgi/>
- Erlang Web Gateway Interface
- Started by Filippo Pacini (sgconsulting.it)
- HTTP server independent
- Based on Python's WSGI (PEP333)
- ... but with an Erlang twist
- Work in progress!
 - If you are pumped, help us

Simple EWGI

```
ewgi_app(Env, StartResp) ->  
  Status = {200, "OK"},  
  Headers = [{"Content-type", "text/plain"}],  
  StartResp(Status, Headers),  
  ["Hello, world!"].
```

EWGI Environment

```
ewgi_app(Env, StartResp) ->  
  Status = {200, "OK"},  
  Headers = [{"Content-type", "text/plain"}],  
  StartResp(Status, Headers),  
  ["Hello, world!"].
```

- Proplist
- HTTP headers
- Functions for additional I/O

EWGI Environment

- CGI-like variables
 - REQUEST_METHOD, SCRIPT_NAME, PATH_INFO, HTTP_ACCEPT, ...
- EWGI variables
 - ewgi.read_input
 - ewgi.write_error
 - ewgi.version
 - ewgi.url_scheme
- Extensible with application-specific variables

EWGI StartResponse

```
ewgi_app(Env, StartResp) ->  
    Status = {200, "OK"},  
    Headers = [{"Content-type", "text/plain"}],  
    StartResp(Status, Headers),  
    ["Hello, world!"].
```

- fun start_response/2
- Arguments: status tuple, response header proplist
- Returns: fun write/1

EWGI Response

```
ewgi_app(Env, StartResp) ->  
  Status = {200, "OK"},  
  Headers = [{"Content-type", "text/plain"}],  
  StartResp(Status, Headers),  
  ["Hello, world!"].
```

- Application returns an IO list
- Or a tuple {iolist(), Thunk}
- Where Thunk returns either a tuple or '\$eof'

Chunked Responses

- Unknown message sizes + persistent connections
 - Long-polling ("Comet")
 - Streaming data
- Done automatically with thunks
 - fun/1 → {Next, Thunk}



Chunked example

```
{<<"This ">>, fun() ->
  {<<"is ">>, fun() ->
    {<<"a ">>, fun() ->
      {<<"chunked ">>, fun() ->
        {<<"response.">>, '$eof'}
      end
    end
  end
end
end
end
```

Middleware

- Behaves like server
 - Wraps application
 - Provides environ and start_response
- Behaves like application
 - Exposed as 2-arity function
 - Follows the same specification
- Composable

Middleware example (simplified)

```
capitalise(App, Env, StartResp) →  
  fun(Env, StartResp) →  
    Res = App(Env, StartResp),  
    lists:map(fun string:to_upper/1,  
              Res)  
  end.
```

Testing

- Unit Testing
 - Create “imitation” requests
 - All applications/middleware follow EWGI spec
- Property-based
 - EWGI is a specification
 - Middleware have properties
 - Applications have properties

Possible future work

- EEP (Erlang Enhancement Proposal)
 - Currently drafting
- Server implementations
 - MochiWeb complete
 - Yaws partially complete
 - Inets, others?

Introducing smak

- <http://github.com/skarab/smak>
- Inspired by Python Paste (Ian Bicking)
- Collection of EWGI middleware
 - Sessions
 - Authentication
 - Caching
- Templates
- URL routing

Templating

- Default is SGTE (SG Templating Engine)
 - Also by Filippo Pacini
 - <http://code.google.com/p/sgte/>
- “Push” data into templates
- Enforces MVC separation
- Cache compiled templates

SGTE Example

```
1> Template = "Hello, $name$!".  
"Hello, $name$!"  
2> {ok, C} = sgte:compile(Template).  
{ok, [<<"Hello, ">>, {attribute, name}, 1], <<"!">>]}  
3> sgte:render_str(C, [{name, "world"}]).  
"Hello, world!"
```

Sessions

- “Share-nothing” cookies
 - Encrypted by Rijndael
- Process per session
 - A la SPEWF
- Ets and Mnesia-based sessions
 - Cookie stores unique hash

Authentication

- Current
 - Bcrypt-hased passwords
- Future
 - Flexible ACL for permissions

Caching

- Obeys HTTP headers
 - Cache-Control
- Different caching backends
 - Memcached
 - Database (PostgreSQL, MySQL)
 - Filesystem
 - Mnesia or ets

Future smak work

- URL routing performance/flexibility
- Scripting for automation
- More middleware components

Thanks!

Hunter Morris

hunter.morris@smarkets.com

<http://www.smarkets.com/>