Getting Some REST with webmachine

Kevin A. Smith

What is webmachine?

Framework

~~Framework~~

Toolkit

A toolkit for building RESTful HTTP resources

What is REST?

Style not a standard

Resources == URLs

http://localhost:8000/hello_world

http://foo.com/hr/emp/123

GET

POST

DELETE

PUT

REST is the shape of the web

```
-module(hello_world_resource).
-export([init/1, to_html/2]).

-include_lib("webmachine/include/webmachine.hrl").

init([]) ->
  {ok, "Hello, world"}.

to_html(Req, State) ->
  Output = io_lib:format("<html><body>~s</body></html>",
                         [State]),
  {Output, Req, State}.
```

```
-module(hello_world_resource).
-export([init/1, to_html/2]).

-include_lib("webmachine/include/webmachine.hrl").

init([]) ->
  {ok, "Hello, world"}.

to_html(Req, State) ->
  Output = io_lib:format("<html><body>~s</body></html>",
                         [State]),
  {Output, Req, State}.
```

```
-module(hello_world_resource).
-export([init/1, to_html/2]).

-include_lib("webmachine/include/webmachine.hrl").

init([]) ->
  {ok, "Hello, world"}.

to_html(Req, State) ->
  Output = io_lib:format(“<html><body>~s</body></html>”,
                         [State]),
  {Output, Req, State}.
```

```
-module(hello_world_resource).
-export([init/1, to_html/2]).

-include_lib("webmachine/include/webmachine.hrl").

init([]) ->
  {ok, "Hello, world"}.

to_html(Req, State) ->
  Output = io_lib:format(“<html><body>~s</body></html>”,
                         [State]),
  {Output, Req, State}.
```

```
-module(hello_world_resource).
-export([init/1, to_html/2]).

-include_lib("webmachine/include/webmachine.hrl").

init([]) ->
  {ok, "Hello, world"}.

to_html(Req, State) ->
  Output = io_lib:format("<html><body>~s</body></html>",
                         [State]),
  {Output, Req, State}.
```

A toolkit for

easily

building RESTful
HTTP
resources

```
GET /test HTTP/1.1
Host: localhost:8000
User-Agent: Mozilla/5.0
Accept: text/html;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: UTF-8,*
Keep-Alive: 300
Connection: keep-alive

HTTP/1.x 200 OK
Server: MochiWeb/1.1 WebMachine/1.3
Date: Mon, 22 Jun 2009 02:27:46 GMT
Content-Type: text/html
Content-Length: 78
```

```
-module(hello_world_resource).
-export([init/1, to_html/2]).
-export([generate_etag/2]).

-include_lib("webmachine/include/webmachine.hrl").

init([]) ->
  {ok, "Hello, world"}.
```

```
to_html(Req, State) ->
  Output = io_lib:format("<html><body>~s</body></html>",
                         [State]),
  {Output, Req, State}.

generate_etag(Req, State) ->
  {mochihex:to_hex(crypto:md5(State)), Req, State}.
```

```
GET /test HTTP/1.1
Host: localhost:8000
User-Agent: Mozilla/5.0
Accept: text/html;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: UTF-8,*
Keep-Alive: 300
Connection: keep-alive

HTTP/1.x 200 OK
Server: MochiWeb/1.1 WebMachine/1.3
Etag: bc6e6f16b8a077ef5fbc8d59d0b931b9
Date: Mon, 22 Jun 2009 02:29:46 GMT
Content-Type: text/html
Content-Length: 78
```

```
GET /test HTTP/1.1
Host: localhost:8000
.
.
.
If-None-Match: bc6e6f16b8a077ef5fbc8d59d0b931b9
```

```
HTTP/1.x 304 Not Modified
Server: MochiWeb/1.1 WebMachine/1.3
Etag: bc6e6f16b8a077ef5fbc8d59d0b931b9
Date: Mon, 22 Jun 2009 02:30:00 GMT
```

```
-module(hello_world_resource).
-export([init/1, to_html/2]).
-export([generate_etag/2, encodings_provided/2]).

-include_lib("webmachine/include/webmachine.hrl").

init([]) ->
  {ok, "Hello, world"}.

to_html(Req, State) ->
  Output = io_lib:format("<html><body>~s</body></html>",
                         [State]),
  {Out, Req, State}.

generate_etag(Req, State) ->
  {mochihex:to_hex(crypto:md5(State)), Req, State}.

encodings_provided(Req, State) ->
  {[{"gzip", fun(X) -> zlib:gzip(X) end}], Req, State}.
```

```
GET /test HTTP/1.1
Host: localhost:8000
.
.
.
Accept-Encoding: gzip,deflate
```

```
HTTP/1.x 200 OK
Server: MochiWeb/1.1 WebMachine/1.3
Etag: bc6e6f16b8a077ef5fbc8d59d0b931b9
Date: Mon, 22 Jun 2009 02:46:57 GMT
Content-Type: text/html
Content-Length: 71
Content-Encoding: gzip
```

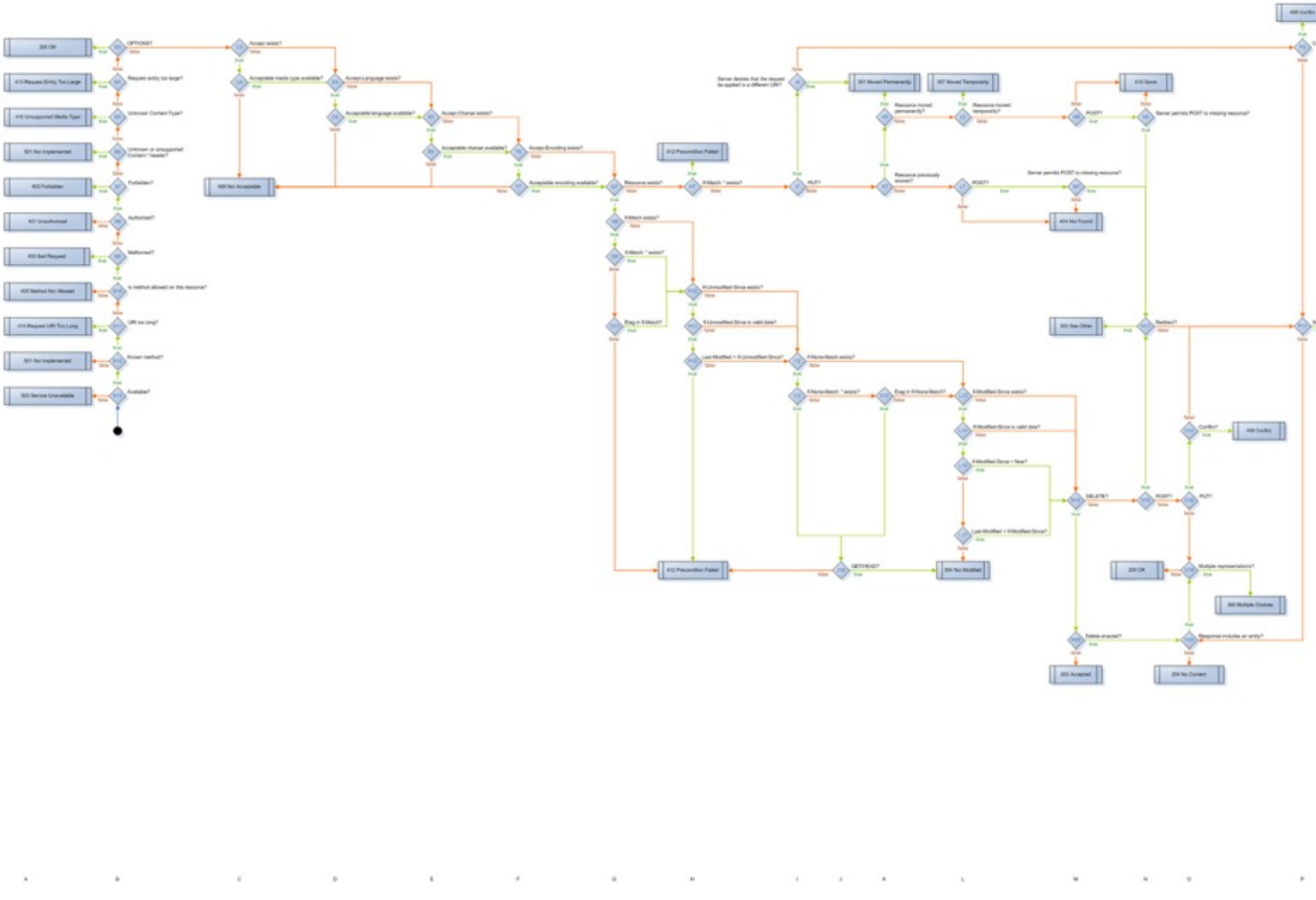
HTTP Is Hard

HTTP/1.1 (DELETE, GET, HEAD, PUT, POST)

An activity diagram to describe the resolution of the response status code, given various headers

Creator: http://thoughtpad.net/alan-dean/
Source: http://thoughtpad.net/alan-dean/http-headers-status

v3



ERROR CODES

PUT *vs.* POST

CONTENT NEGOTIATION

ENCODINGS

IDEMPOTENCY

CONTENT EXPIRATION

STREAMING

OVERLOADED POST*s*

CONDITIONAL GET

Request

Routing

http://foo.com/items

```
{["items"], grocery_item_resource, []}.
```

URL path

Resource module

Init params

http://foo.com/items/chocolate

```
{["items", item], grocery_item_resource, []}.
```

URL path

URL variable

Resource module

Init params

GET

```
-module(hello_world_resource).
-export([init/1,to_html/2]).

-include_lib("webmachine/include/webmachine.hrl").

init([]) ->
  {ok, "Hello, world"}.

to_html(Req, State) ->
  Output = io_lib:format(“<html><body>~s</body></html>”,
                         [State]),
  {Output, Req, State}.
```

Must be idempotent

PUT

```
-module(grocery_list_resource).
-export([init/1,allowed_methods/2]).
-export([content_types_accepted/2,from_json/2]).

-include_lib("webmachine/include/webmachine.hrl").

init([]) ->
  {ok, []}.
```

```
allowed_methods(Req, State) ->
  {['PUT'], Req, State}.

content_types_accepted(Req, State) ->
  {[{"application/json", from_json}], Req, State}.

from_json(Req, State) ->
  %% Create/update logic here
```

DELETE

```
-module(grocery_list_resource).
-export([init/1,allowed_methods/2]).
-export([delete_resource/2]).

-include_lib("webmachine/include/webmachine.hrl").

init([]) ->
  {ok, []}.
```

```
allowed_methods(Req, State) ->
  {['DELETE'], Req, State}.

delete_resource(Req, State) ->
  %% Deletion logic here
```

POST

Hmmmm

Problems with POST

- Overloaded semantics
- Create or update?

Creation via POST

- `allowed_methods/2`
- `post_is_create/2`
- `create_path/2`
- `content_types_accepted/2`
- handler function

Update via POST

- `allowed_methods/2`
- `content_types_accepted/2`
- handler function

CODE
TIME

HTTP Request Access

HTTP Request

- "Wrapped" mochiweb request
- Separate process for each request
- Direct access to: request/response headers, response body, etc.

Other Cool Stuff

- Graphical request debugging
- Streaming requests and responses
- File uploads

webmachine source:

http://bitbucket.org/justin/webmachine

Slides and demo code:

http://github.com/kevsmith/erlang_factory/tree/master