Building an API with Mojolicious

Florent Mara
<florent.mara@gmail.com>

Context

- ‘Incoming XML payload from the interweb’

- Node.js API
- No pre-existing web framework
- Isolation required
  - jump host
  - inline credentials
  - no access to other subsystems
- Looking for a blueprint

Why not extend existing Node.js API?

- It is performant but …
- High TCO (module/AWS/security …)
- No authentication
- No tests
- Currently internal
- Not isolated
- Not conducive to feature scaling or code complexity
- Limited in-house skills
- … but it is performant

Why Mojolicious?

- The website looks good
- It says it is modern
- Many blog posts document success using Mojolicious
- Documentation is clear and indicates a planned progression from Mojolicious::Lite to Mojolicious
- The tutorial was simple and worked flawlessly
- It is Perl
- Likely tests will be possible, potentially even easy

Why not Dancer? Catalyst? Other Web Framework?

- Dunno

Getting started

- $ cpanm -i Mojolicious
- $ mojo generate lite_app myapp.pl

```
#!/usr/bin/env perl
use Mojolicious::Lite;

# Documentation browser under "/perldoc"
plugin 'PODRenderer';

get '/' => sub {
  my $c = shift;
  $c->render(template => 'index');
};

app->start;
__DATA__

@@ index.html.ep
% layout 'default';
% title 'Welcome';
<h1>Welcome to the Mojolicious real-time web framework!</h1>
To learn more, you can browse through the documentation
<%= link_to 'here' => '/perldoc' %>.

@@ layouts/default.html.ep
<!DOCTYPE html>
<html>
  <head><title><%= title %></title></head>
  <body><%= content %></body>
</html>

```

What happened?

- One file was created
- Uses packaged Mojolicious::Lite
- Sets plugin PODRenderer
- Sets an HTTP get method on path ‘/’ that renders a template
- Calls app->start
- Defines the template

```
$ ./myapp.pl daemon
Server available at http://127.0.0.1:3000
```

Great ...

- … for rapid prototyping

- Limited maintainability / reusability / extensibility
- Code and template within file.
- No MVC

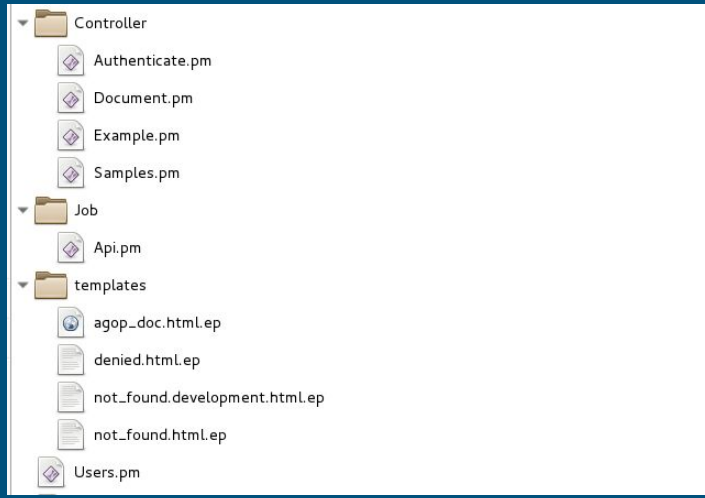
Mojolicious

- $ mojo generate app my_app.pl

```
myapp                    # Application directory
|- script                # Script directory
|  +- my_app             # Application script
|- lib                   # Library directory
|  |- MyApp.pm           # Application class
|  +- MyApp              # Application namespace
|     +- Controller      # Controller namespace
|        +- Example.pm   # Controller class
|- t                     # Test directory
|  +- basic.t            # Random test
|- log                   # Log directory
|  +- development.log    # Development mode log file
|- public                # Static file directory (served automatically)
|  +- index.html         # Static HTML file
+- templates             # Template directory
   |- layouts            # Template directory for layouts
   |  +- default.html.ep # Layout template
   +- example            # Template directory for "Example" controller
      +- welcome.html.ep # Template for "welcome" action
```

Our Mojolicious App (1/5)

- /lib and /t common to all projects -> top level
- /script -> /Job
- /Controller and /templates relocated
- No use for /public
- Config file is ‘somewhere up’
- .pl is ‘somewhere upper’




```
#!/usr/bin/perl

use 5.020;

use FindBin;
use lib "$FindBin::Bin/../lib";

use Mojolicious::Commands;
Mojolicious::Commands->start_app('Mojo::Job::Api', 'daemon', '-l', 'http://*:8080');
```

Our Mojolicious App: api.pm (2/5)

```
package Mojo::Job::Api;
use Mojo::Base 'Mojolicious';

use File::Find::Rule;
use FindBin;

use Mojo::Log;
use Mojolicious::Plugin::Config;

sub startup {
    my $self = shift;

    # Initialization

    # Initialize routes and controller

    # Set default routes

    # Authentication routes

    # Documentation

    # The actual work to get done

}
```

Our Mojolicious App: Initialization (3/5)

```
# Initialization
my $purpose   = 'agop';
my $prefix_01 = $purpose.'/0.1'; # API version number

my @conf_array = File::Find::Rule->file->name('mojo_*.conf')->in($FindBin::Bin.'/..');
unless( defined($conf_array[0]) ) { die 'No conf file found ... dying now ... arghhhhh ..... A!'; }

$self->plugin('PODRenderer');
$self->plugin('Config', { file => $conf_array[0] });
$self->mode($self->config('environment'));
$self->secrets(['                    ','                    ','                    ']);

$self->renderer->paths([$FindBin::Bin.'/../lib/Mojo/templates']);

my $log = Mojo::Log->new(path => '/var/log/mojo.log', level => 'info');
$log->info('Mojo project started');
```

Our Mojolicious App: Config file (4/5)

```
{ name        => 'Mojo',
  vendors     => { testoptics => 'secret_auth_key', },
  environment => 'development',
  timezone    => 'Pacific/Auckland',
  s3_bucket   => 'our_s3_bucket', };
```

Our Mojolicious App: Routes (5/5)

```
# Initialize routes and controller
my $r = $self->routes();
push( @{$self->routes->namespaces}, 'Mojo::Controller');

# Set default routes
$r->any('/')        ->to( controller => 'example', action => 'index');
$r->any('/welcome')->to( controller => 'example', action => 'welcome');
$r->any('/ping')    ->to( controller => 'example', action => 'ping');
$r->any('/error')   ->to( controller => 'example', action => 'error');

# Authentication routes
$r->any('/login') ->to( controller => 'authenticate', action => 'login', prefix => $prefix_01, log => $log, );
$r->any('/logout')->to( controller => 'authenticate', action => 'logout', log => $log, );

my $auth = $r->under('/')->to( controller => 'authenticate', action => 'logged_in');

# Documentation
$auth->any($prefix_01.'/doc')->to( controller => 'document', action => 'doc');

# The actual work to get done
$r->post($prefix_01.'/samples')->to( controller => 'samples', action => 'save');
```

Some testing

- use Test::Mojo;

```
# Given ... a test Mojo object.
my $t = Test::Mojo->new($object);

# Given ... looking for root path
# When ... Then ...
$t->get_ok('/')->status_is(302);

# Given ... looking for welcome path
# When ... Then ...
$t->get_ok('/welcome')->status_is(200)->content_is('Welcome to Regen API');

# Given ... looking for ping path
# When ... Then ...
$t->get_ok('/ping')->status_is(200)->content_is('1');

# Given ... looking for error path
# When ... Then ...
$t->get_ok('/error')->status_is(404)->content_is('Not Found');
$t->get_ok('/error?code=401&message=goodmessage')->status_is(401)->content_is('goodmessage');
```

For more information

http://mojolicious.org/