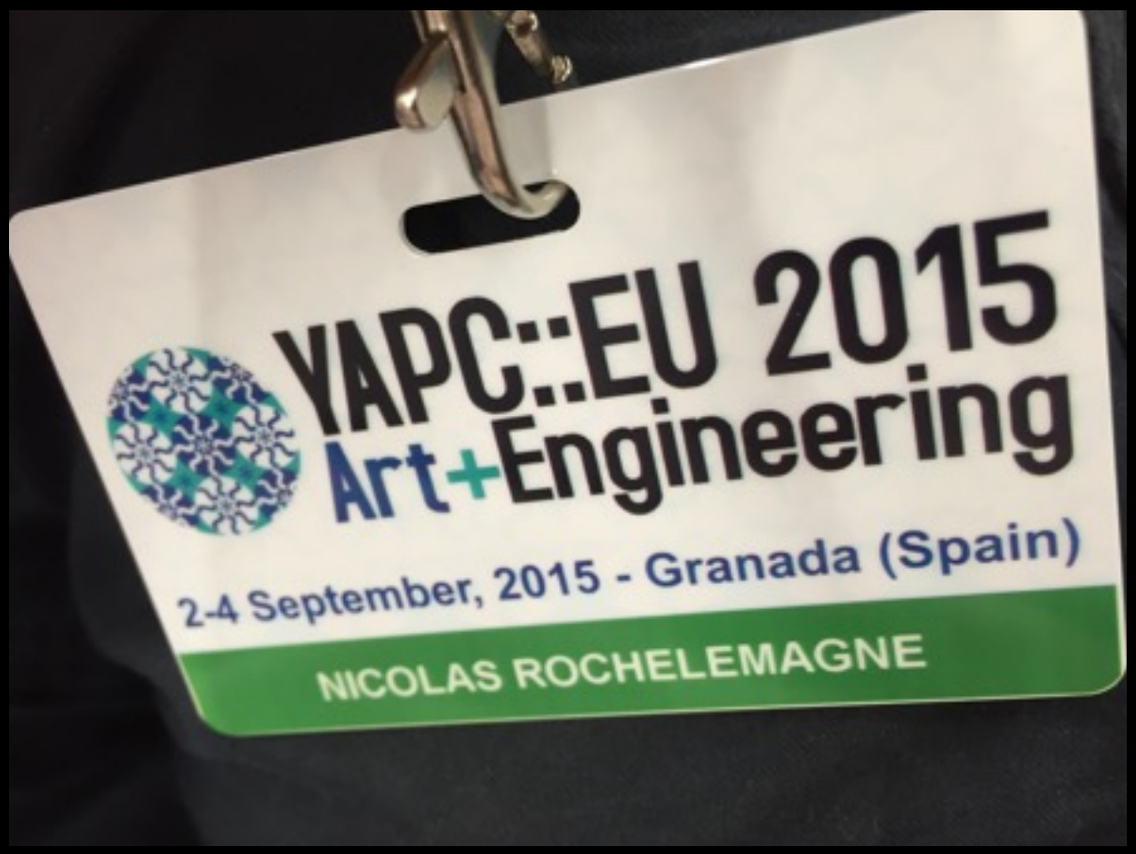


From: *YAPC::EU@Houston.pm*

From: YAPC::EU@Houston.pm

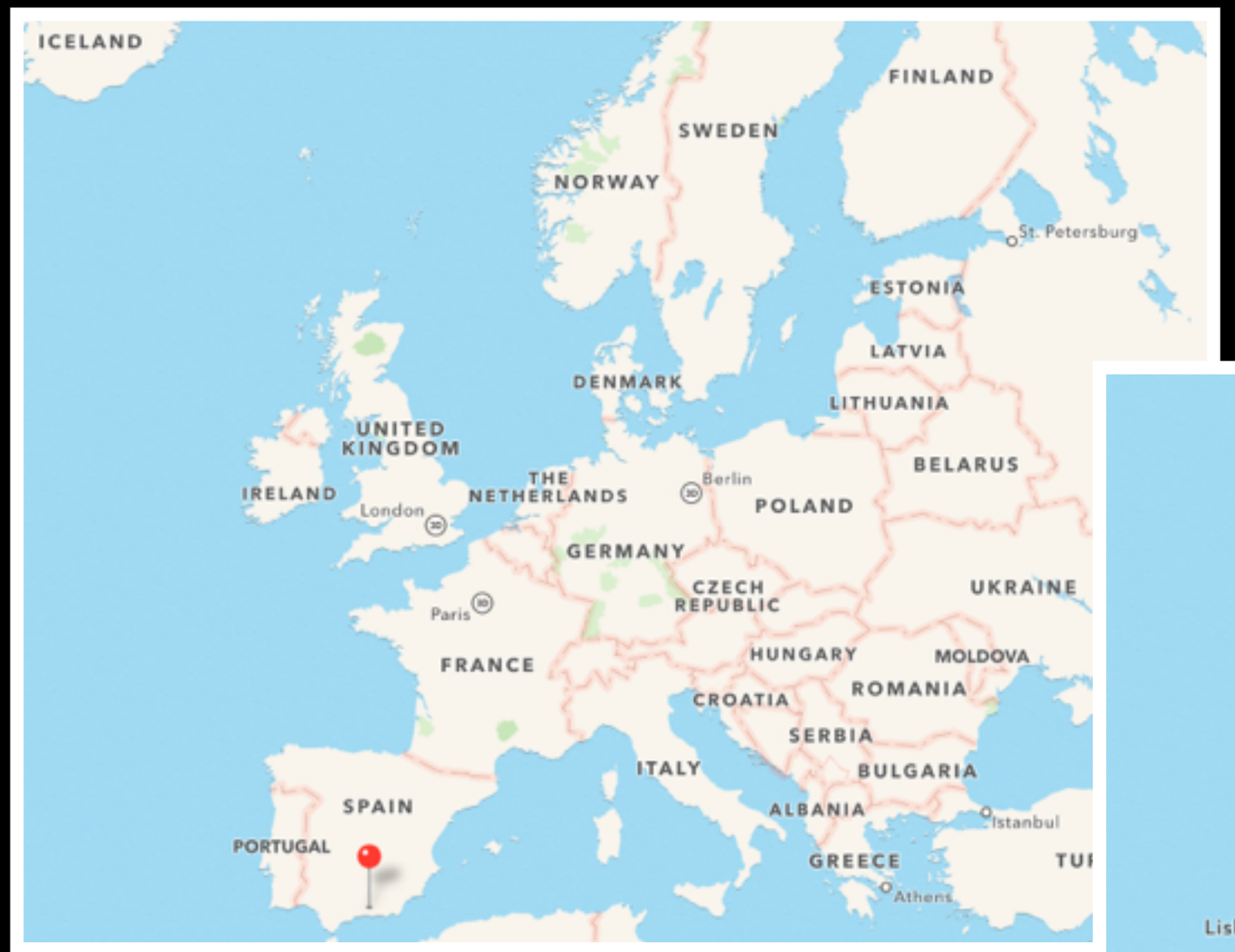
1. Web Scrapers
2. Platform incompatibilites
3. Compiler status
4. WebService::Simple
5. Server::Starter
6. Starlet
7. Cookie::Baker
8. Gazelle
9. Furl
10. Riki
11. Perl::Lint
12. Carmel

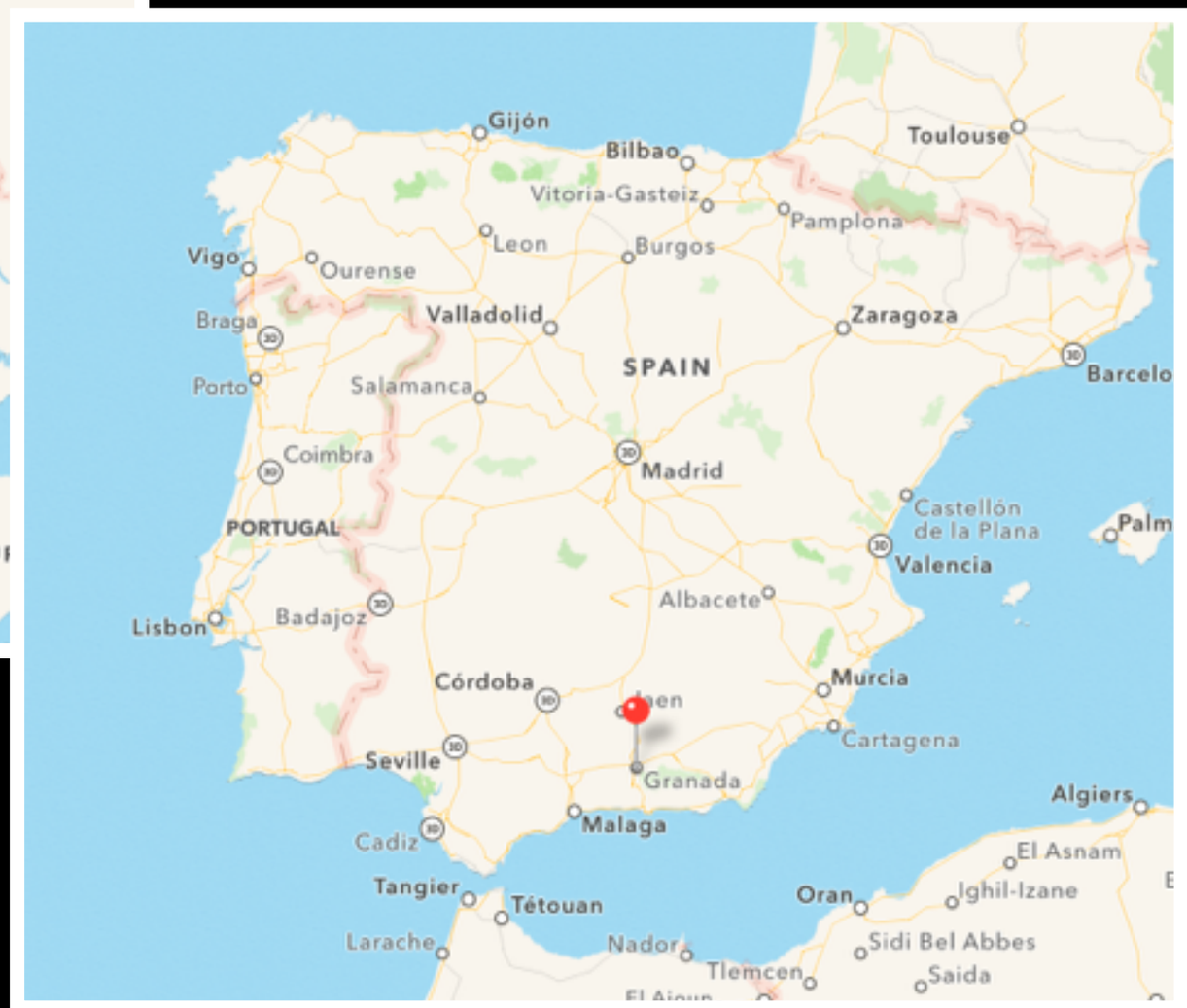
Granada ?


Gulf of Mexico
BAHAMAS
Havana
CUBA
EXICO
Guadalajara
lexico City
Puebla
Acapulco
JAMAICA
DOMINICAN REPUBLIC
San Juan
BELIZE
GUATEMALA
Guatemala City
HONDURAS
NICARAGUA
COSTA RICA
PANAMA
Maracaibo
Caracas
Port of Spain
VENEZUELA
Medellin
Bogotá
Santiago De Cali
COLOMBIA
Georgetown
GUYANA
FRENCH GUIANA

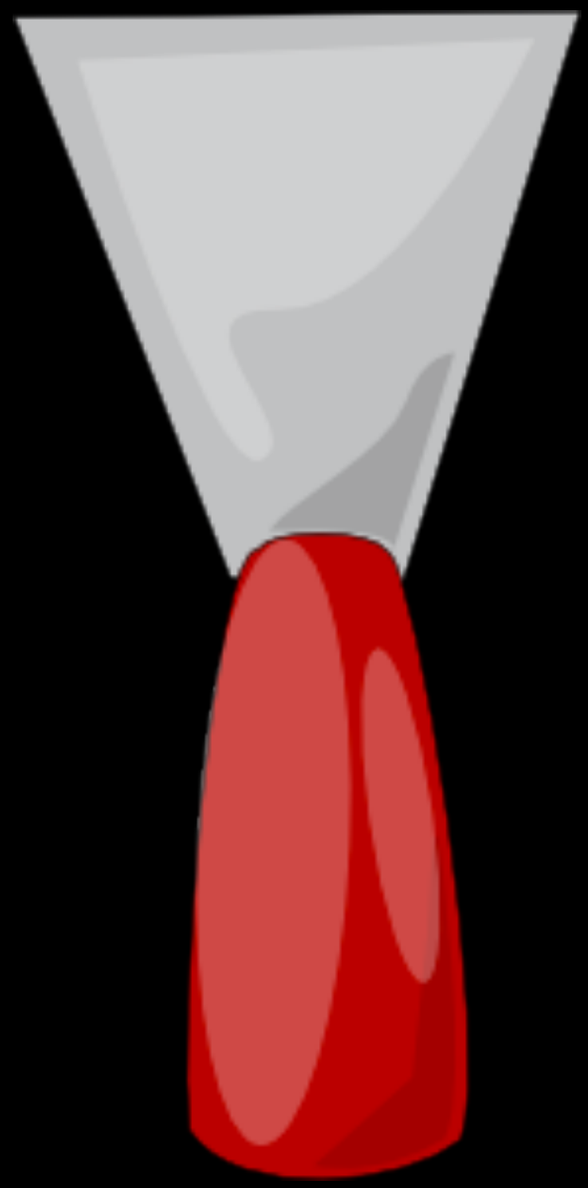

SAINT BARTHÉLEMY
asseterre
St. John's
MONTSERRAT
GUADELOUPE
DOMINICA
Fort-de-France
SAINT LUCIA
SAINT VINCENT AND THE GRENADINES
Bridgetown
GRENADA
Port of Spain
TRINIDAD AND TOBAGO

Granada, Spain






PERL



Web Scrapers

from @Sawyer

Web Scrapers

- **What is it ?**
  application to extract content from a webpage….

1. Crawl
2. Parse
3. Extract content

- **But Why ?**

1. *Website without API…*
2. *Website with an API…*

Html code can change…
but api also…

Web Scrapers

Scrapers for dummies

```perl
my $ua = LWP::UserAgent->new;
my $response = $ua->get('http://search.cpan.org/');

if ($response->is_success) {
    print $response->decoded_content;  # or whatever
}
else {
    die $response->status_line;
}
```

Web Scrapers

Two tools for scrapping:

- WWW::Mechanize
  browser in an object
  forms, links, “back button”,
  “reload button”

- Web::Query
  selector based

Web Scrapers: WWW::Mechanize

```perl
my $mech = WWW:Mechanize->new;
$mech->agent_alias( q{Windows IE 6} )

my $content = $mech->get($url)
$mech->follow_link( url_regex => qr/download/i );
$mech->reload();
# from LWP::UserAgent
$mech->post( $url, { email => q{you@Houston.pm}, ... })
$mech->submit_form(
    form_name => 'search',
    fields    => { query  => 'can you find it' },
    button    => 'Search Now'
);
$mech->back();
```

Web Scrapers:

Web::Query *by Tokuhiro Matsuno*

```
<div class="head">
    <dt>Some text</dt>
    <dt>to test</dt>
</div>
```

```
wq($url)->find(q/div.head dt/)
    ->each( sub {
        my $ix = shift;
        say $_->text # WQ object
        }
    );
```

Platform incompatibilities

Talk by Mark Overmeer
author of POSIX::1003

- POSIX 1003.1: attempt to standardize the operating systems since 1984:
  libraries: libc, libc, …
  OS: I/O, events, permissions, memory,…

- over 1200 functions

POSIX is difficult

> perldoc -f print

```
print FILEHANDLE LIST
print FILEHANDLE
print LIST
print   Prints a string or a list of strings.  Returns true if
        successful.  FILEHANDLE may be a scalar variable containing the
```

POSIX.pm is old

> perldoc POSIX

NOTE

The POSIX module is probably the most complex Perl module supplied with the standard distribution. It incorporates autoloading, namespace games, and dynamic loading of code that's in Perl, C, or both. It's a great source of wisdom.

POSIX is weak

Sample with select

> perldoc -f select

select FILEHANDLE
select RBITS, WBITS, EBITS, TIMEOUT

> perldoc POSIX::1003::Events

```
select(RBITS, WBITS, EBITS, [TIMEOUT])
    Perl core contains two functions named "select".  The second is the
    one we need here.  Without TIMEOUT, the select will wait until an
    event emerges (or an interrupt).
```

```
sub select($$$;$)
{   push @_, undef if @_==3;
    goto &select;
}
```

POSIX: other issues

- getpid => $$ # 12 simple rewrites
- POSIX::open != open| # 57 name clashes
- 20 croak “use method xxx instead"
- 59 croak “xxx is C-specific use yyy"

POSIX::1003

```
POSIX/1003
├── Confstr.pm
├── Confstr.pod
├── Errno.pm
├── Errno.pod
├── Events.pm
├── Events.pod
├── FS.pm
├── FS.pod
├── Fcntl.pm
├── Fcntl.pod
├── FdIO.pm
├── FdIO.pod
├── Limit.pm
├── Limit.pod
├── Locale.pm
├── Locale.pod
├── Math.pm
├── Math.pod
├── Module.pm
├── Module.pod
├── OS.pm
├── OS.pod
├── Pathconf.pm
├── Pathconf.pod
├── Proc.pm
├── Proc.pod
├── Properties.pm
├── Properties.pod
├── Signals.pm
├── Signals.pod
├── Symbols.pm
├── Sysconf.pm
├── Sysconf.pod
├── Termios.pm
├── Termios.pod
├── Time.pm
├── Time.pod
├── User.pm
└── User.pod
```

DESCRIPTION

The POSIX::1003 suite implements access to many POSIX functions. The POSIX module in core (distributed with Perl itself) is ancient, the documentation is usually wrong, and it has too much unusable code in it. "POSIX::1003" tries to provide cleaner access to the operating system. More about the choices made can be found in section "Rationale".

POSIX::1003

Exporter trick: +1

```
If your import list starts with +1, the symbols will not get into your
own namespace, but that of your caller. Just like
$Exporter::ExportLevel (but a simpler syntax).
```

Sample

```
sub MyModule::import(@)   # your own tricky exporter
{   POSIX::1003->import('+1', @_);
}
```

…what's the trick ?

POSIX::1003

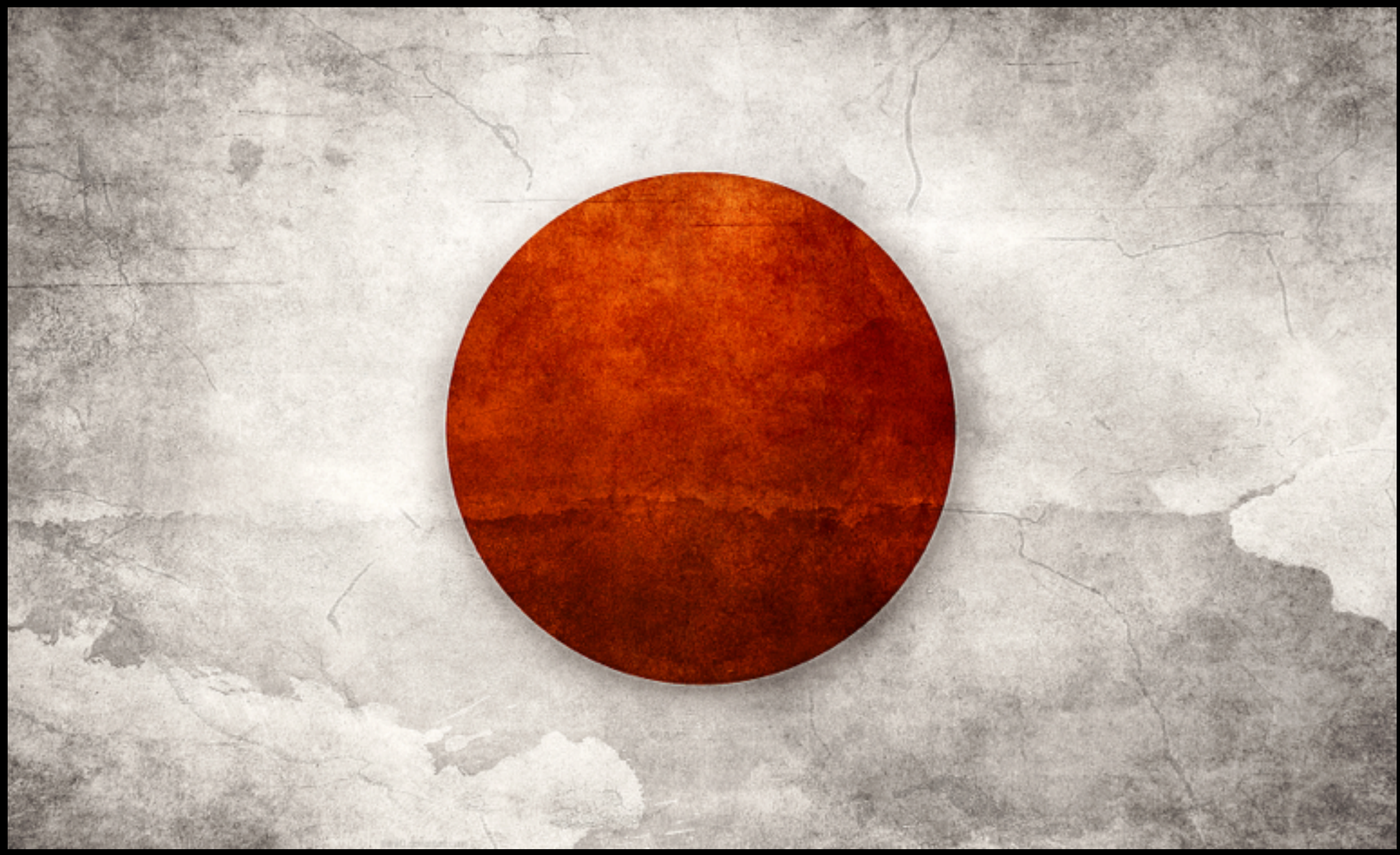
```
# extract from POSIX/1003.pm
 sub import(@)
 {    my $class = shift;

      my $level = @_ && $_[0] =~ /^\+(\d+)$/ ? shift : 0;
      return if @_ == 1 && $_[0] eq ':none';
      @_ ||= ':all';

      no strict   'refs';
      no warnings 'once';
      my $to    = (caller $level)[0];

      foreach (@_) {
         ...
         *{$to.'::'.$_} = \&$_;
         ...
      }
```

Compiler status

perl compiler 5.14.4 in production since 2014
@cPanel [CentOS 5,6,7]

Advantages:

- 10 to 20 % less memory usage
- faster startup
- no need to ship your modules / code
  (dynamic modules are loaded on demand)



Junichi Ishida aka uzulla

slides: http://bit.ly/uzulla_yapceu2015

“Why does it need to be so fast ? because we are…”

Japanese

WebService::Simple

```perl
use WebService::Simple;

# Simple use case
my $flickr = WebService::Simple->new(
    base_url => "http://api.flickr.com/services/rest/",
    param    => { api_key => "your_api_key", }
);

# send GET request to
# http://api.flickr.com/service/rest/?
# api_key=your_api_key&method=flickr.test.echo&name=value
$flickr->get( { method => "flickr.test.echo", name => "value" }

# send GET request to
# http://api.flickr.com/service/rest/extra/path?
#     api_key=your_api_key&method=flickr.test.echo&name=value
$flickr->get( "extra/path",
              { method => "flickr.test.echo", name => "value" });

# ->post
# can enable cache
my $cache = Cache::File->new(..., default_expires => '30 min' );
WebService::Simple->new( ..., cache => $cache );
```

Server::Starter

- extra protection layer on top of your plack server
- "a superdaemon for hot-deploying server programs"
- Gracefull restart (hot-deploying)
- Only exit old process if new process is successfully booted.(safe!)
- No resource leak (known)

```
> start_server \
  --interval 5 \
  --port 8000 \
  --signal-on-hup=QUIT \ # for Starman
  -- \
  starman --preload-app myapp.psgi
```

Starlet

- "a simple, high-performance PSGI/Plack HTTP server"
- Very heavily used in japan.

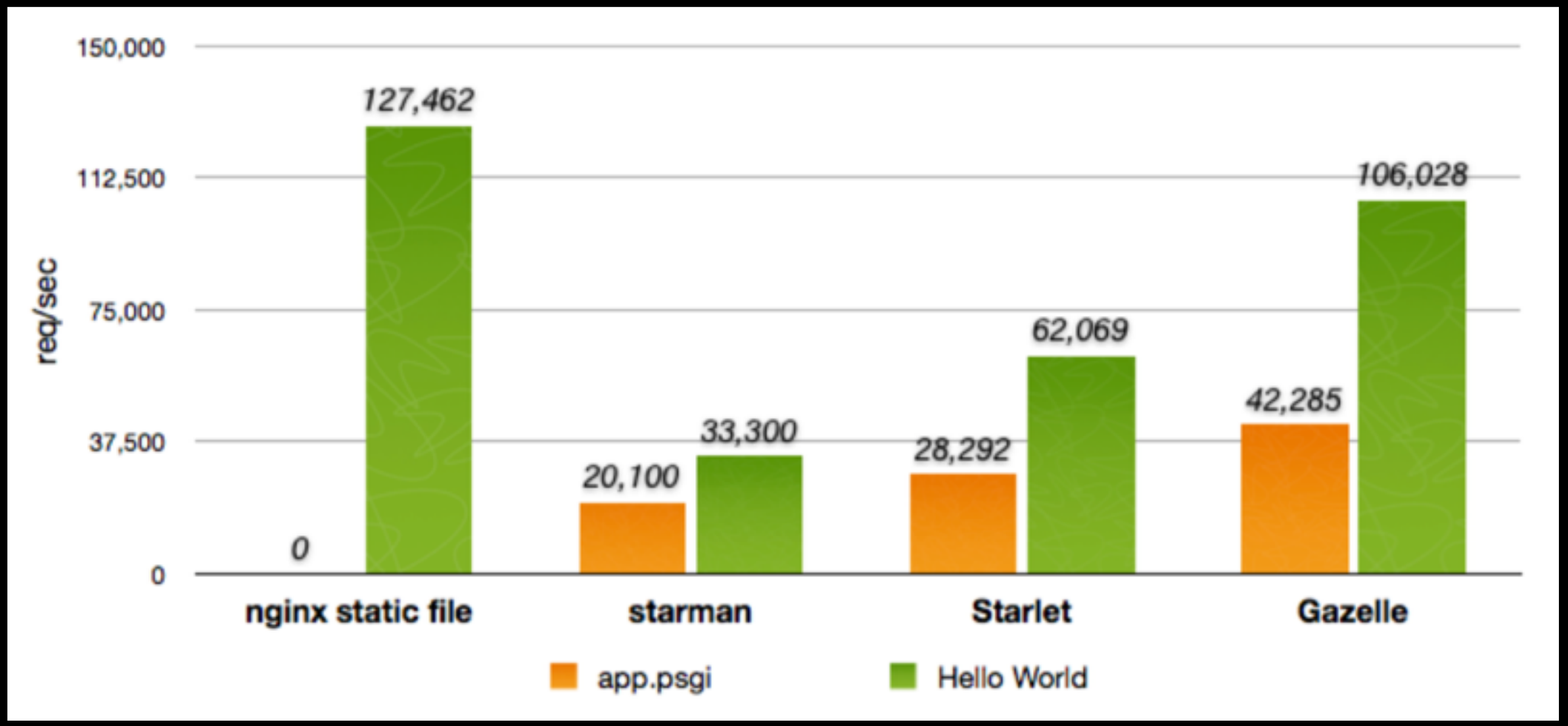
```
$ plackup -s Starlet app.psgi
$ start_server --port=8000 -- \
    plackup -s Starlet app.psgi

# some available parameters for performance tuning.
--max-workers=#
--timeout=#
--keepalive-timeout=#
--max-keepalive-reqs=#
--max-reqs-per-child=#
--min-reqs-per-child=#
--spawn-interval=#
```

Cookie::Baker

```
# bake a cookie...
my $cookie = bake_cookie('foo','val');
my $cookie = bake_cookie('foo', {
    value => 'val',
    path => "test",
    domain => '.example.com',
    expires => '+24h'
} );

$headers->push_header('Set-Cookie', bake_cookie($key,$val));

# ... or crush a few of them
my $cookies_hr = crush_cookie($headers->header('Cookie'));
```

Gazelle

a *very* fast{/jp} PSGI server



x2 faster than starman

Furl

simple & fast http client [made in Japan]

```
use Furl;
my $furl = Furl->new(
    agent   => 'MyGreatUA/2.0',
    timeout => 10,
);
my $res = $furl->get('http://example.jp/');
print $res->content;
```

Furl - post sample

```
use Furl;
my $furl = Furl->new(
    agent   => 'MyGreatUA/2.0',
    timeout => 10,
);
my $res = $furl->post(
    'http://example.jp/', # URL
    [ q/X-MY-HEADER/ =>'ohmy' ], # headers
    # form data (HashRef/FileHandle are also okay)
    [ foo => 'bar' ],
);
print $res->content;
```

Riji - git based blog tool

```
$ cpanm Riji
$ mkdir some_dir; cd some_dir
$ riji setup
$ vi article/entry/start.md
$ git add . ; git commit -a
$ riji server
$ open http://localhost:3650/entry/start.html
```

publish [static] files

```
# edit blog meta data once.(author, title...)
$ vi riji.yml
# generate htmls in blog dir
$ riji publish
```

Perl::Lint *by Kawakami*

Why ?

- "Yet Another Perl Source Code Linter"
- faster than other lint tools
- fast and flexible static analyzer for Perl5
- compatibility with Perl::Critic
- easy to extend/add policies

```
                Rate Perl::Critic   Perl::Lint
Perl::Critic 20.6/s            --         -78%
Perl::Lint   92.0/s          348%           --
```

Perl::Lint *by Kawakami*

```perl
package Perl::Lint::Policy::Subroutines::ProhibitExplicitReturnUndef;
use strict;
use warnings;
use Perl::Lint::Constants::Type;
use parent "Perl::Lint::Policy";

use constant {
    DESC => '"return" statement with explicit "undef"',
    EXPL => [199],
};

sub evaluate {
    my ($class, $file, $tokens, $args) = @_;

    my @violations;
    for (my $i = 0; my $token = $tokens->[$i]; $i++) {
        my $token_type = $token->{type};

        if ($token_type == RETURN) {
            my $next_token = $tokens->[++$i];
            if ($next_token->{type} == DEFAULT && $next_token->{data} eq 'undef') {
                    push @violations, {
                        filename => $file,
                        line     => $token->{line},
                        description => DESC,
                        explanation => EXPL,
                        policy => __PACKAGE__,
                    };
                    next;
            }
        }
    }

    return \@violations;
}

1;
```

Carmel (or Carton 2)

Talk from Tatsuhiko Miyazawa (>200 CPAN modules):

- cpanminus
- carton
- Corona
- PSGI
- Plack
- Starman
- Twiggy
- …

Carmel (or Carton 2)

Carton with cache to speedup

cache blib in a common directory
$HOME/.carmel/{version}-{archname}/builds

```
$HOME/.carmel/5.20.1-darwin-2level/builds
  Plack-1.0033/
    blib/
      arch/
      lib/
  URI-1.64/
    blib/
      arch/
      lib/
  URI-1.63/
    blib/
      arch/
      lib/
```

Carmel (or Carton 2)

```
# cpanfile + cpanm

> cat cpanfile
require Test::More;
require FindBin;
requires 'DBD::SQLite', '== 1.36_04';
requires 'Dancer', '== 1.150';

# <MEANING>                  <EXPRESION>                 <EXAMPLE>
# equal                      == VERSION                   == 0.03
# greater than or equal      VERSION or >= VERSION       0.03 or >= 0.03
# lower than or equal        <= VERSION                  <= 0.03
# greater than               > VERSION                   > 0.03
# lower than                 < VERSION                   < 0.03

> perl cpanm --installdeps .
> perl cpanm -L extlib --installdeps .
```

```
# Run with a directory with cpanfile
carmel install

# Manually pull a module if you don't have it
carmel inject DBI@1.633 Plack@1.0000

# list all the modules to be loaded
carmel list

# list all the modules in a tree
carmel tree

# show a location where a module is installed
carmel show Plack

# Runs your perl script with modules from artifacts
carmel exec perl ...

# Requires all your modules in cpanfile in one shot
carmel exec perl -e 'use Carmel::Preload;'

# Roll out the currently selected modules into ./local
carmel rollout

# package modules tarballs and index into ./vendor/cache
carmel package

# use Carmel packages inside a script (without carmel exec)
perl -e 'use Carmel::Setup; ...'

# prints export PATH=... etc for shell scripting
carmel export

# find a module in a repository
carmel find DBI

# find a module matching the version query
carmel find Plack ">= 1.0000, < 1.1000"
```

```
# cpanfile + carmel

> carmel install
Using DBD::SQLite (1.36_04)
Using DBI (1.634)
---> Installing new dependencies: Dancer
Successfully installed HTTP-Server-Simple-0.50
Successfully installed HTTP-Server-Simple-PSGI-0.16
Successfully installed URI-1.69
Successfully installed LWP-MediaTypes-6.02
Successfully installed Encode-Locale-1.05
Successfully installed IO-HTML-1.001
Successfully installed HTTP-Date-6.02
Successfully installed HTTP-Message-6.10
Successfully installed HTTP-Body-1.22
Successfully installed MIME-Types-2.11
Successfully installed Devel-StackTrace-2.00
Successfully installed Class-Data-Inheritable-0.08
Successfully installed Exception-Class-1.39
Successfully installed Dancer-1.150
14 distributions installed
---> Complete! 2 cpanfile dependencies. 16 modules installed.
---> Use `carmel show [module]` to see where a module is installed.
```

```
> carmel list
Class::Data::Inheritable (0.08)
DBD::SQLite (1.36_04)
DBI (1.634)
Dancer (1.150)
Devel::StackTrace (2.00)
Encode::Locale (1.05)
Exception::Class (1.39)
HTTP::Body (1.22)
HTTP::Date (6.02)
HTTP::Message (6.10)
HTTP::Server::Simple (0.50)
HTTP::Server::Simple::PSGI (0.16)
IO::HTML (1.001)
LWP::MediaTypes (6.02)
MIME::Types (2.11)
URI (1.69)
```

```
> carmel tree
DBD::SQLite (1.36_04)
 DBI (1.634)
Dancer (1.150)
 Exception::Class (1.39)
  Class::Data::Inheritable (0.08)
  Devel::StackTrace (2.00)
 HTTP::Body (1.22)
  HTTP::Message (6.10)
   Encode::Locale (1.05)
   HTTP::Date (6.02)
   IO::HTML (1.001)
   LWP::MediaTypes (6.02)
   URI (1.69)
 HTTP::Server::Simple::PSGI (0.16)
  HTTP::Server::Simple (0.50)
MIME::Types (2.11)
```

```
> carmel show Dancer
Dancer (1.150) in /Users/nicolas/.carmel/5.14.4-darwin-2level/builds/Dancer-1.15(

> perl -E 'use Carmel::Setup; use Dancer; say $INC{"Dancer.pm"}'
/Users/nicolas/.carmel/5.14.4-darwin-2level/builds/Dancer-1.150/blib/lib/Dancer.p

> perl -E 'use Dancer; say $INC{"Dancer.pm"}'
/Users/nicolas/perl5/lib/perl5//Dancer.pm

> carmel export
export PERL5OPT="-MCarmel::Setup" PERL_CARMEL_PATH="/Users/nicolas/workspace/gitl
```

```
# try/upgrade to a new version

> grep Dancer cpanfile
#requires 'Dancer', '== 1.150';
requires 'Dancer', '== 1.3111';

> carmel install
Using DBD::SQLite (1.36_04)
Using DBI (1.634)
---> Installing new dependencies: Dancer
Successfully installed Try-Tiny-0.22
Successfully installed File-Listing-6.04
Successfully installed HTTP-Negotiate-6.01
Successfully installed HTML-Tagset-3.20
Successfully installed HTML-Parser-3.71
Successfully installed HTTP-Daemon-6.01
Successfully installed Net-HTTP-6.09
Successfully installed HTTP-Cookies-6.01
Successfully installed WWW-RobotRules-6.02
Successfully installed libwww-perl-6.13
Successfully installed Dancer-1.3111 (upgraded from 1.150)
11 distributions installed
---> Complete! 2 cpanfile dependencies. 23 modules installed.
---> Use `carmel show [module]` to see where a module is installed.

> carmel show Dancer
Dancer (1.3111) in /Users/nicolas/.carmel/5.14.4-darwin-2level/builds/Dancer-1.3111

> perl -E 'use Carmel::Setup; use Dancer; say $INC{"Dancer.pm"}'
/Users/nicolas/.carmel/5.14.4-darwin-2level/builds/Dancer-1.3111/blib/lib/Dancer.pm
```

```
> carmel find Dancer
Dancer (1.3111) in /Users/nicolas/.carmel/5.14.4-darwin-2level/builds/Dancer-1.3111
Dancer (1.150) in /Users/nicolas/.carmel/5.14.4-darwin-2level/builds/Dancer-1.150

> carmel tree
DBD::SQLite (1.36_04)
 DBI (1.634)
Dancer (1.3111)
 HTTP::Body (1.22)
  HTTP::Message (6.10)
   Encode::Locale (1.05)
   HTTP::Date (6.02)
   IO::HTML (1.001)
   LWP::MediaTypes (6.02)
   URI (1.69)
HTTP::Server::Simple::PSGI (0.16)
 HTTP::Server::Simple (0.50)
LWP (6.13)
 File::Listing (6.04)
 HTML::Parser (3.71)
  HTML::Tagset (3.20)
 HTTP::Cookies (6.01)
 HTTP::Daemon (6.01)
 HTTP::Negotiate (6.01)
 Net::HTTP (6.09)
 WWW::RobotRules (6.02)
MIME::Types (2.11)
Try::Tiny (0.22)
```

```
> carmel rollout
Installing DBD-SQLite-1.36_04 to /Users/nicolas/workspace/github/Simple-Accessor/local
Installing DBI-1.634 to /Users/nicolas/workspace/github/Simple-Accessor/local
Installing Dancer-1.3111 to /Users/nicolas/workspace/github/Simple-Accessor/local
Installing HTTP-Body-1.22 to /Users/nicolas/workspace/github/Simple-Accessor/local
Installing HTTP-Message-6.10 to /Users/nicolas/workspace/github/Simple-Accessor/local
Installing Encode-Locale-1.05 to /Users/nicolas/workspace/github/Simple-Accessor/local
Installing HTTP-Date-6.02 to /Users/nicolas/workspace/github/Simple-Accessor/local
Installing IO-HTML-1.001 to /Users/nicolas/workspace/github/Simple-Accessor/local
Installing LWP-MediaTypes-6.02 to /Users/nicolas/workspace/github/Simple-Accessor/local
Installing URI-1.69 to /Users/nicolas/workspace/github/Simple-Accessor/local
Installing HTTP-Server-Simple-PSGI-0.16 to /Users/nicolas/workspace/github/Simple-Accessor/local
Installing HTTP-Server-Simple-0.50 to /Users/nicolas/workspace/github/Simple-Accessor/local
Installing libwww-perl-6.13 to /Users/nicolas/workspace/github/Simple-Accessor/local
Installing File-Listing-6.04 to /Users/nicolas/workspace/github/Simple-Accessor/local
Installing HTML-Parser-3.71 to /Users/nicolas/workspace/github/Simple-Accessor/local
Installing HTML-Tagset-3.20 to /Users/nicolas/workspace/github/Simple-Accessor/local
Installing HTTP-Cookies-6.01 to /Users/nicolas/workspace/github/Simple-Accessor/local
Installing HTTP-Daemon-6.01 to /Users/nicolas/workspace/github/Simple-Accessor/local
Installing HTTP-Negotiate-6.01 to /Users/nicolas/workspace/github/Simple-Accessor/local
Installing Net-HTTP-6.09 to /Users/nicolas/workspace/github/Simple-Accessor/local
Installing WWW-RobotRules-6.02 to /Users/nicolas/workspace/github/Simple-Accessor/local
Installing MIME-Types-2.11 to /Users/nicolas/workspace/github/Simple-Accessor/local
Installing Try-Tiny-0.22 to /Users/nicolas/workspace/github/Simple-Accessor/local

> perl -E 'use Carmel::Setup; use Dancer; say $INC{"Dancer.pm"}'
local/lib/perl5/Dancer.pm
```

```
> tree local
local
├── bin
│   ├── dancer
│   ├── dbilogstrip
│   ├── dbiprof
│   ├── dbiproxy
│   ├── lwp-download
│   ├── lwp-dump
│   ├── lwp-mirror
│   └── lwp-request
└── lib
    └── perl5
        ├── Dancer
        │   ├── App.pm
        │   ├── Config
        │   │   └── Object.pm
        │   ├── Config.pm
        │   ├── Continuation
        │   │   ├── Halted.pm
        │   │   ├── Route
.....

> rm -Rf local/
> perl -E 'use Carmel::Setup; use Dancer; say $INC{"Dancer.pm"}'
/Users/nicolas/.carmel/5.14.4-darwin-2level/builds/Dancer-1.3111/blib/lib/Dancer.pm
```

```
# restore previous module version

> vi cpanfile # restore previous version
> carmel install
Using DBD::SQLite (1.36_04)
Using DBI (1.634)
Using Dancer (1.150)
Using Exception::Class (1.39)
Using Class::Data::Inheritable (0.08)
Using Devel::StackTrace (2.00)
Using HTTP::Body (1.22)
Using HTTP::Message (6.10)
Using Encode::Locale (1.05)
Using HTTP::Date (6.02)
Using IO::HTML (1.001)
Using LWP::MediaTypes (6.02)
Using URI (1.69)
Using HTTP::Server::Simple::PSGI (0.16)
Using HTTP::Server::Simple (0.50)
Using MIME::Types (2.11)
---> Complete! 2 cpanfile dependencies. 16 modules installed.
---> Use `carmel show [module]` to see where a module is installed.
> perl -E 'use Carmel::Setup; use Dancer; say $INC{"Dancer.pm"}'
/Users/nicolas/.carmel/5.14.4-darwin-2level/builds/Dancer-1.150/blib/lib/Dancer.pm
```

...and also

- Benchmark::Perl::Formance
- Eixo::Zone: interact with (OS) namespaces
- JWT tokens - *http://jwt.io*
  - JSON::WebToken
  - Crypt::JWT
- Monitoring graphs
  - GrowthForecast (needs rrdtool)
  - HRForecast (no rrdtool required
- Test::mysqld
- Test::Valgrind
  valgrind frontend: memory errors & leaks
- *mRuby is mruby binding for perl5*
- Measure::Everything
  provides a standard measuring API for modules
- Menlo: cpanminus 2
- WebService::Mackerel - client for mackerel.io (performance monitor tool on cloud)

Thank you !