Dancer
(the Effortless Perl Web Framework)
About me

- Sawyer X
- Sysadmin / Perl Ninja
- Israel.pm / Haifa.pm / TelAviv.pm / Rehovot.pm
- I do system, networking, web, applications, etc.
- [http://blogs.perl.org/users/sawyer_x/](http://blogs.perl.org/users/sawyer_x/)
- [http://search.cpan.org/~xsawyerx/](http://search.cpan.org/~xsawyerx/)
Perl web recap

1995

CGI
Perl web recap

2010

Many frameworks

(including micro-frameworks like Dancer)
The big web religions, illustrated
Ruby – the fanboys
Python – the sticklers
PHP – the nonsensical
Perl – the nutcases
Nutcases?

- Yes, we are insane (but not LISP-insane)
- Insanity is a whole lot of fun!
- Insanity gives us flexibility
- Flexibility gives us cool stuff
- Like Moose and meta-programming
- Like DBIx::Class
- Like Dancer
from flask import Flask

app = Flask(__name__)

@app.route("/", methods=['GET'])
def hello():
    return "Hello World!"

if __name__ == "__main__":
    app.run()
Dancer (Perlesque)

use Dancer;

get "/hi" => sub {
    "Hello, World!"
};

dance;
In comparison

```python
from flask import Flask

app = Flask(__name__)

@app.route("/", methods=['GET'])
def hello():
    return "Hello World!"

if __name__ == "__main__":
    app.run()
```

```perl
use Dancer;

get "/" => sub {
    "Hello, World!"
};

dance;
```
Dancer treats

- Both read and write, easily!
- Route-based (started as a port of Sinatra)
- PSGI/Plack compliant (PSGI is our WSGI)
- Minimum dependencies
- Any app is also a web server
- CPAN-friendly (<3 CPAN)
Recipe for Dancing

- Take an HTTP method
- Add a path to that
- Mix with a subroutine
- And sprinkle plugins and keywords on top
Dancer route structure

get  '/path'  =>  sub  {  ...  };
post '/path'  =>  sub  {  ...  };
put  '/path'  =>  sub  {  ...  };
del '/path'  =>  sub  {  ...  };
options '/path'  =>  sub  {  ...  };
any  '/path'  =>  sub  {  ...  };
Dancer

- Paths can contain variables
  
  ```
  get '/hello/:entity/
  ```

- Paths can be Regular Expressions
  
  ```
  get qr{ /
    \(\w+\)
    / \d\{2,3\} \(.+\)? \}x
  ```
Dancer login example

post '/login' => sub {

    # Validate the username and password
    if ( params->{user} eq 'bob' &&
         params->{pass} eq 'LetMeIn' ) {

        session user => params->{user};
        redirect params->{path} || '/';
    } else {
        redirect '/login?failed=1';
    }
};
get '//' => sub {

    template index => {

        greeting => 'welcome'

    }

};
More nifty stuff

- `headers 'My-X-Header' => 'Value'
- `send_file('report.tar.gz')`
- `set_cookie name => 'value',
  expires => ( time + 3600 ),
  domain => 'foo.com'
- `status 'not_found'
- `to_json, to_yam1, to_xml`
- `my $file = upload('file_input')`
- `my $all_upuploads = request->uploads`
Dancer as Perl philosophy

- Dancer is succinct, efficient and easy to work with
- Dancer is daring
  (Do you have route caching in Django?)
  (Websockets in near future!)
- Dancer has a lot of plugins:
  (engines for sessions, logging, templates)
- Serializers (JSON, YAML, XML)
- Route filters (before, after, before_template)
Oh yeah, route caching...
get '/user/:id.:format' => sub {
  UserRS->find( params->{id} );
};

# curl http://mywebservice/user/42.json
{
  "id": 42, "name": "John Foo",
  "email": "john.foo@hopkins.com"
}

# curl http://mywebservice/user/42.yml
--

id: 42
name: "John Foo"
email: "john.foo@hopkins.com"
use Dancer::Plugin::SiteMap;

- You get: /sitemap and /sitemap.xml
- “Yup, it’s that simple.”
Dancer::Plugin::Email

post '/contact' => sub {
    email {
        to => 'a@b.com',
        subject => 'Test',
        message => $msg,
        attach => [ path => 'name' ],
    }
};
post '/login' => sub {
    my $user = params->{'user'};
    my $pass = params->{'pass'};
    if ( auth( $user, $pass ) ) {
        if ( auth_asa('guest') ) {...}
        if ( auth_can('create') ) {...}
    }
};
Dancer::Plugin::Ajax

ajax '/check_for_update' => sub {
  # some ajax code
}

- Pass if X-Request-With not "XMLHttpRequest"
- Disable the layout
- The action built is a POST request
Dancer::Plugin::DBIC

- DBIC (DBIx::Class) – a sophisticated ORM
- Configure the connection in the config file
- Make the ResultSets available in routes
Database(s) connection in Dancer

get '/widget/view/:id' => sub {
    my $sth = database->prepare('select * from widgets where id = ?',
                               'select * from widgets where id = ?');
    $sth->execute(params->{id});
    template display_widget => {
        widget => $sth->fetchrow_hashref,
    };
};
In culmination

Dancer is beautiful and fun
The way programming should be

Perldancer.org
search.cpan.org/perldoc?Dancer