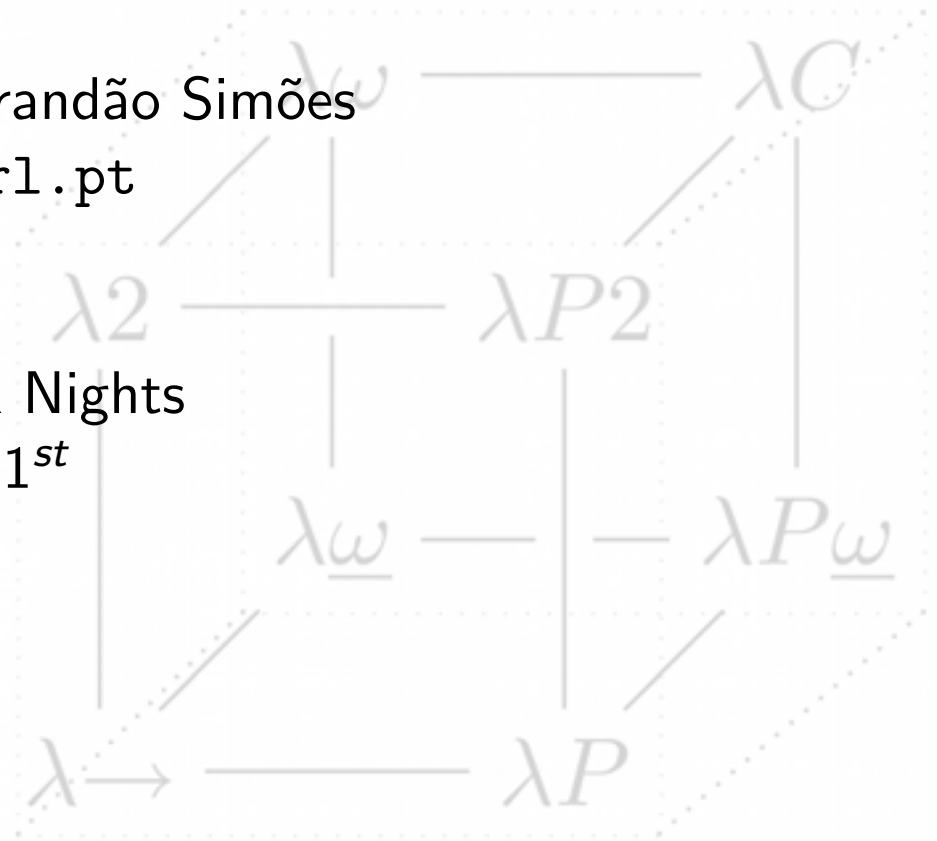
Dancing Tutorial

Alberto Manuel Brandão Simões
ambs@perl.pt

Braga Geek Nights
March 1st



Part 0: Ballet

What's Dancer?

A micro framework
for writing web applications

So, Yet Another Web Framework?

Well, yes!

CGI.pm webapps are spaghetti

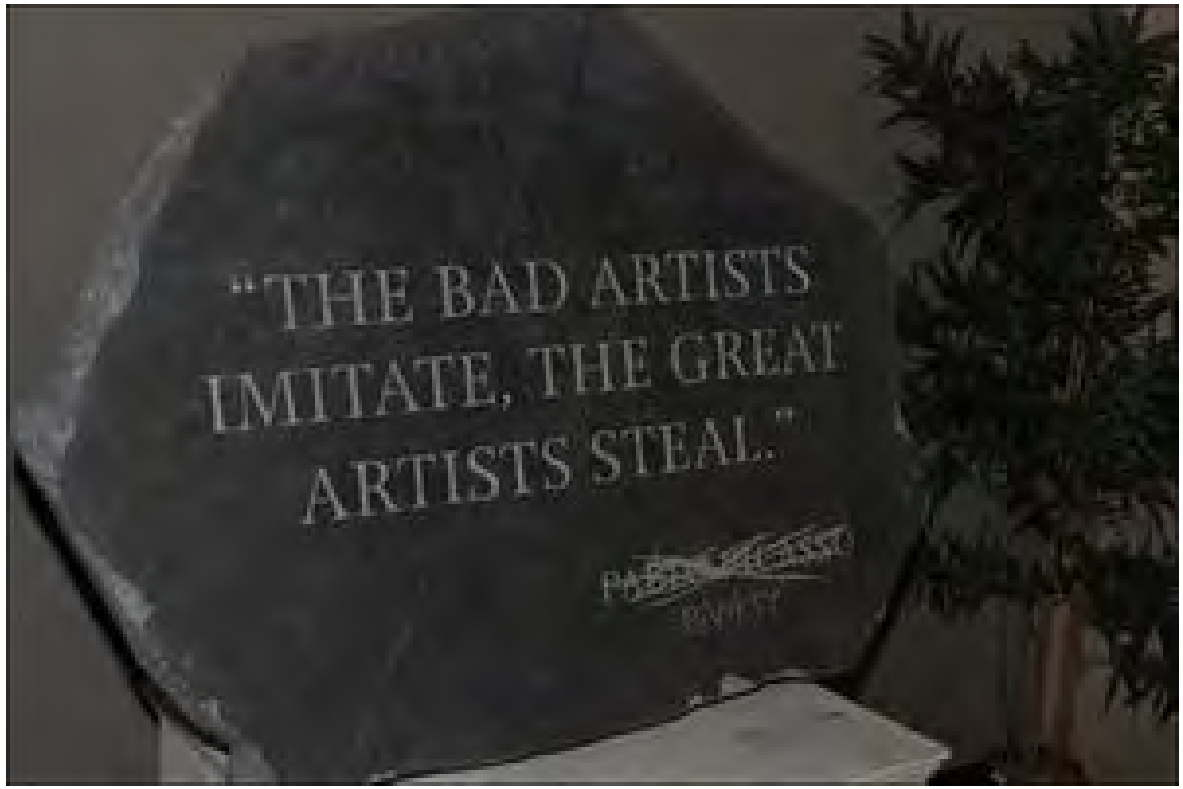
Catalyst is HUGE

Ruby introduced something different


FRANK
Sinatra

We needed something similar in Perl

Definitely



And they also imitated us!

Available at

http://perldancer.org

and in GitHub.

Part 1: Tango

Installing

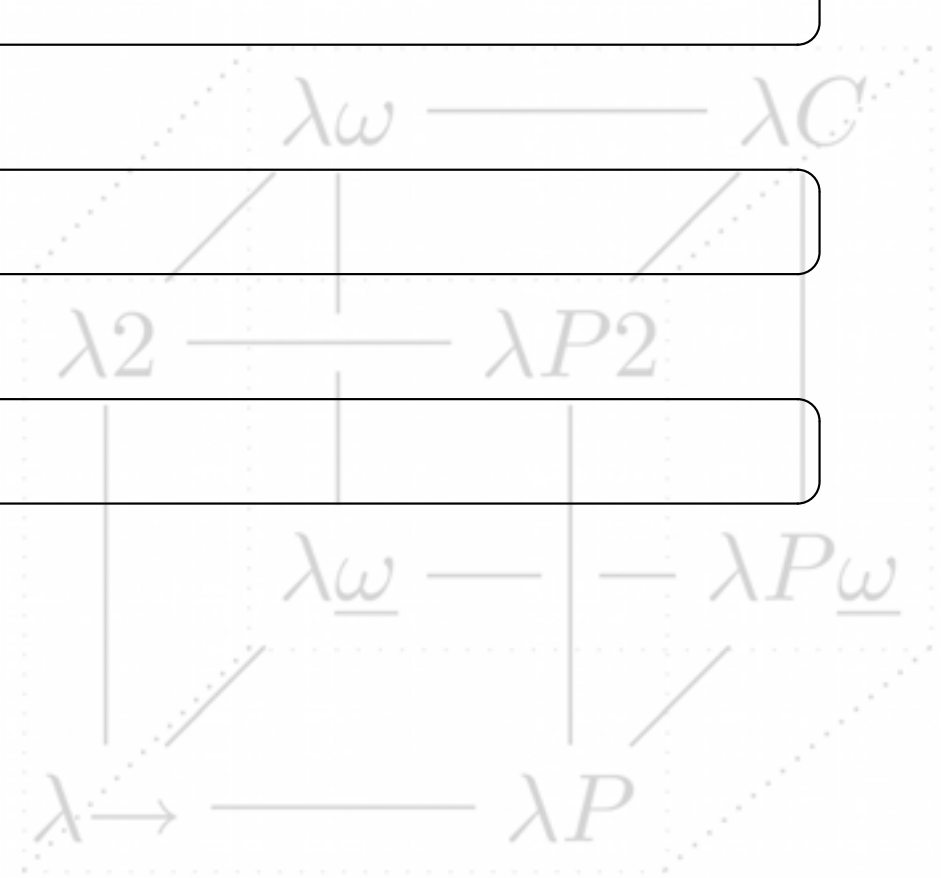
```
$ sudo cpan Dancer
```

or

```
$ sudo cpan Task::Dancer
```

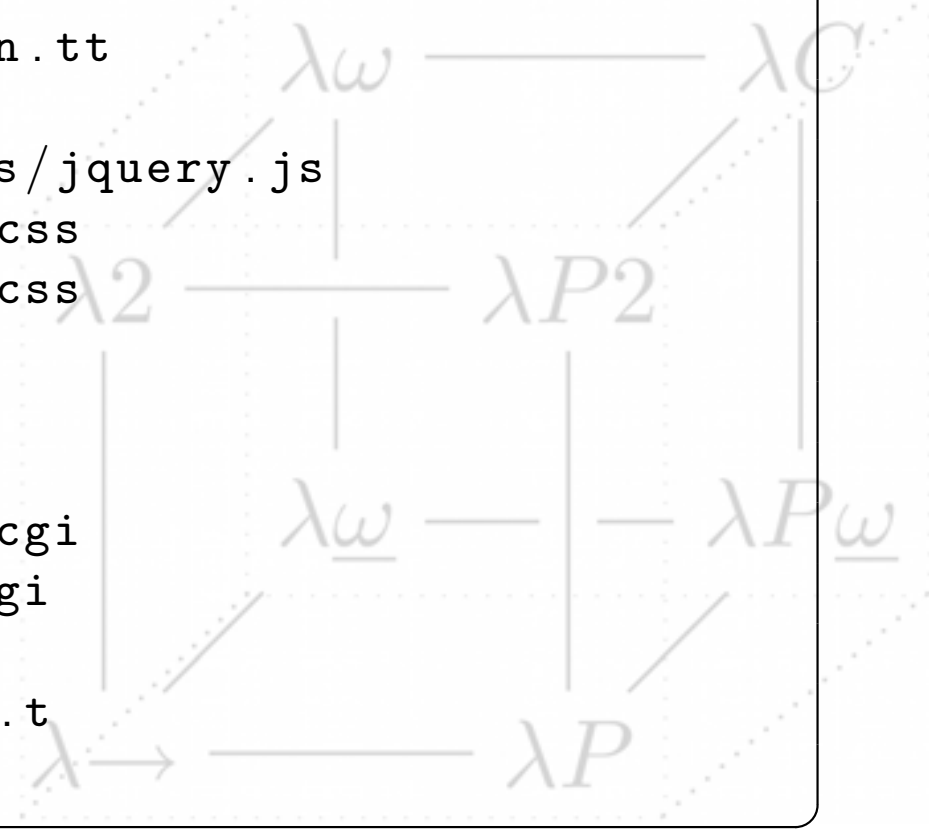
or yet

```
$ cpanm -S Task::Dancer
```



Bootstrap application

```
$ dancer -a MyApp
+ MyApp/bin/app.pl
+ MyApp/config.yml
+ MyApp/environments/development.yml
+ MyApp/environments/production.yml
+ MyApp/views/index.tt
+ MyApp/views/layouts/main.tt
+ MyApp/lib/MyApp.pm
+ MyApp/public/javascripts/jquery.js
+ MyApp/public/css/style.css
+ MyApp/public/css/error.css
+ MyApp/public/images/...
+ MyApp/public/500.html
+ MyApp/public/404.html
+ MyApp/public/dispatch.fcgi
+ MyApp/public/dispatch.cgi
+ MyApp/Makefile.PL
+ MyApp/t/002_index_route.t
+ MyApp/t/001_base.t
```



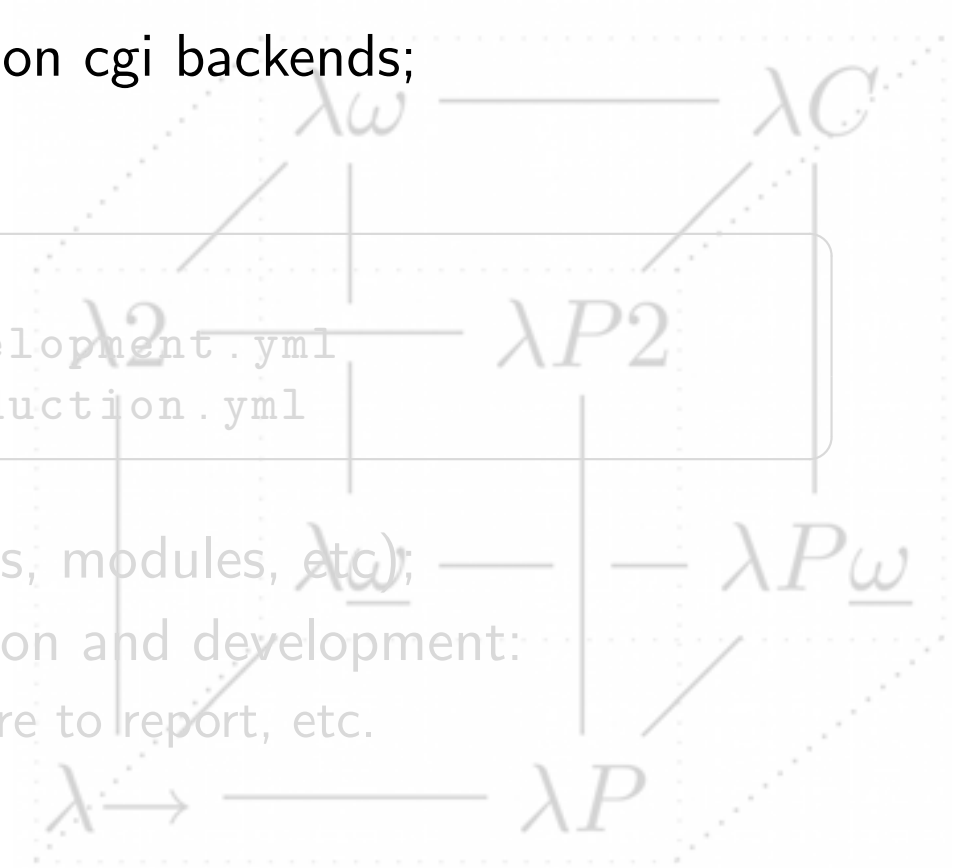
What's inside

```
+ MyApp/bin/app.pl
```

- a standalone light server (starts server in port 3000);
- also used for fast-cgi or common cgi backends;

```
+ MyApp/config.yml
+ MyApp/environments/development.yml
+ MyApp/environments/production.yml
```

- main configuration file (plugins, modules, etc);
- configuration files for production and development:
  - defines what to report, where to report, etc.



What's inside

```
+ MyApp/bin/app.pl
```

- a standalone light server (starts server in port 3000);
- also used for fast-cgi or common cgi backends;

```
+ MyApp/config.yml
+ MyApp/environments/development.yml
+ MyApp/environments/production.yml
```

- main configuration file (plugins, modules, etc);
- configuration files for production and development:
  - defines what to report, where to report, etc.

What's inside

```
+ MyApp/views/index.tt
+ MyApp/views/layouts/main.tt
```

- Templates and layouts:
  - templates are page portions/scraps;
  - layouts are full page designs
    (they are automagically filled by the template);

```
+ MyApp/public/javascripts/jquery.js
+ MyApp/public/css/style.css
+ MyApp/public/css/error.css
+ MyApp/public/images/...
```

- public/static files:
  - javascript (Dancer ships with jsquery);
  - cascade style sheets;
  - images (for default design);

What's inside

```
+ MyApp/views/index.tt
+ MyApp/views/layouts/main.tt
```

- Templates and layouts:
  - templates are page portions/scraps;
  - layouts are full page designs
    (they are automagically filled by the template);

```
+ MyApp/public/javascripts/jquery.js
+ MyApp/public/css/style.css
+ MyApp/public/css/error.css
+ MyApp/public/images/...
```

- public/static files:
  - javascript (Dancer ships with jsquery);
  - cascade style sheets;
  - images (for default design);

What's inside

```
+ MyApp/public/500.html
+ MyApp/public/404.html
```

- pages for 500 and 404 errors;

```
+ MyApp/public/dispatch.fcgi
+ MyApp/public/dispatch.cgi
```

- wrappers to configure fast-cgi and cgi backends;
  - will be back on this later (future talk?);

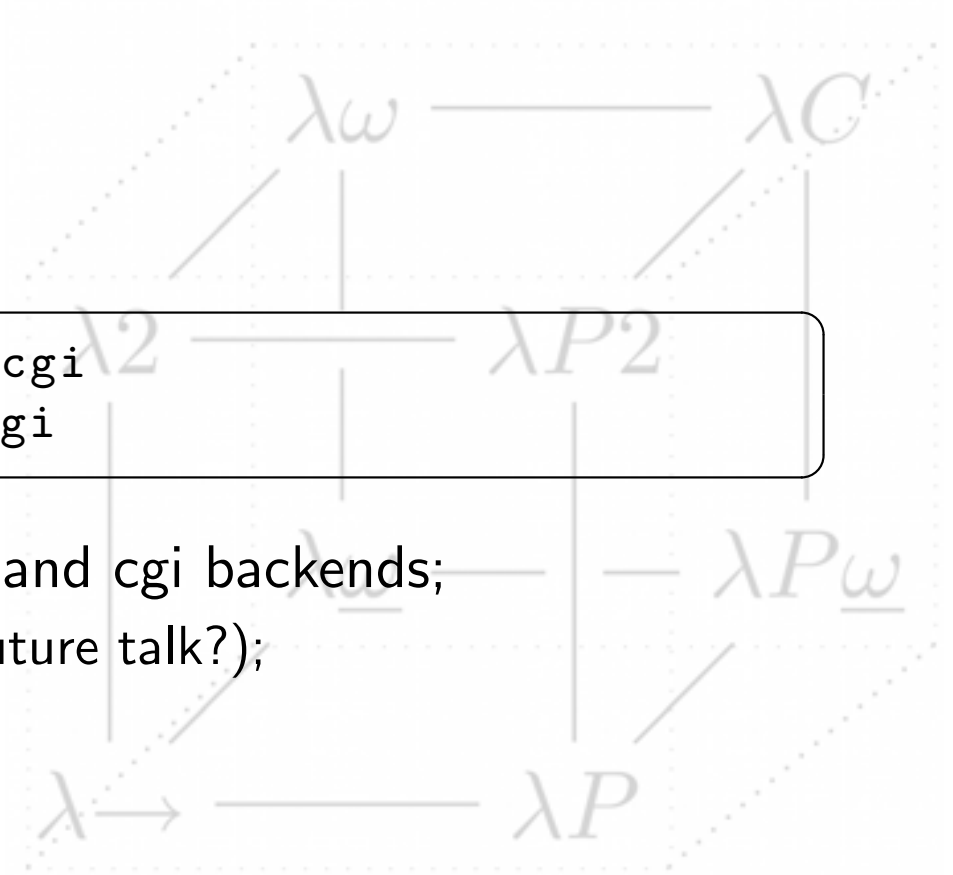
What's inside

```
+ MyApp/public/500.html
+ MyApp/public/404.html
```

- pages for 500 and 404 errors;

```
+ MyApp/public/dispatch.fcgi
+ MyApp/public/dispatch.cgi
```

- wrappers to configure fast-cgi and cgi backends;
  - will be back on this later (future talk?);



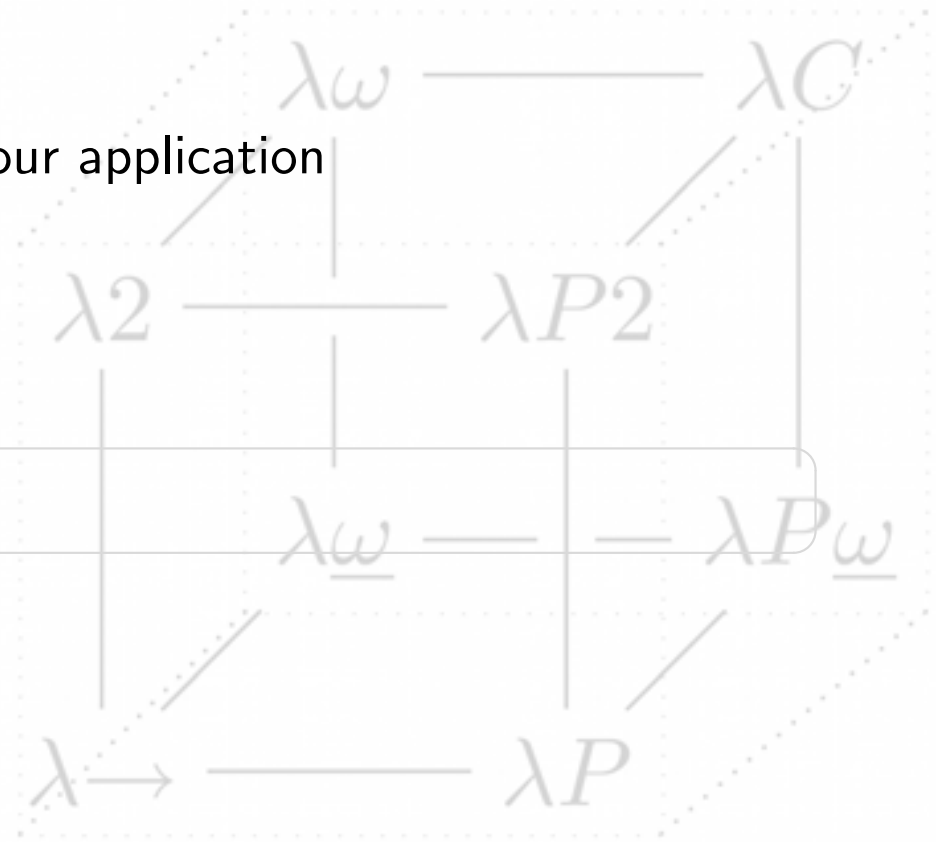
What's inside

```
+ MyApp/Makefile.PL
+ MyApp/t/002_index_route.t
+ MyApp/t/001_base.t
```

- Main module Makefile:
  - usefull to make module of your application
- Your test suite;

```
+ MyApp/lib/MyApp.pm
```

- Your application!



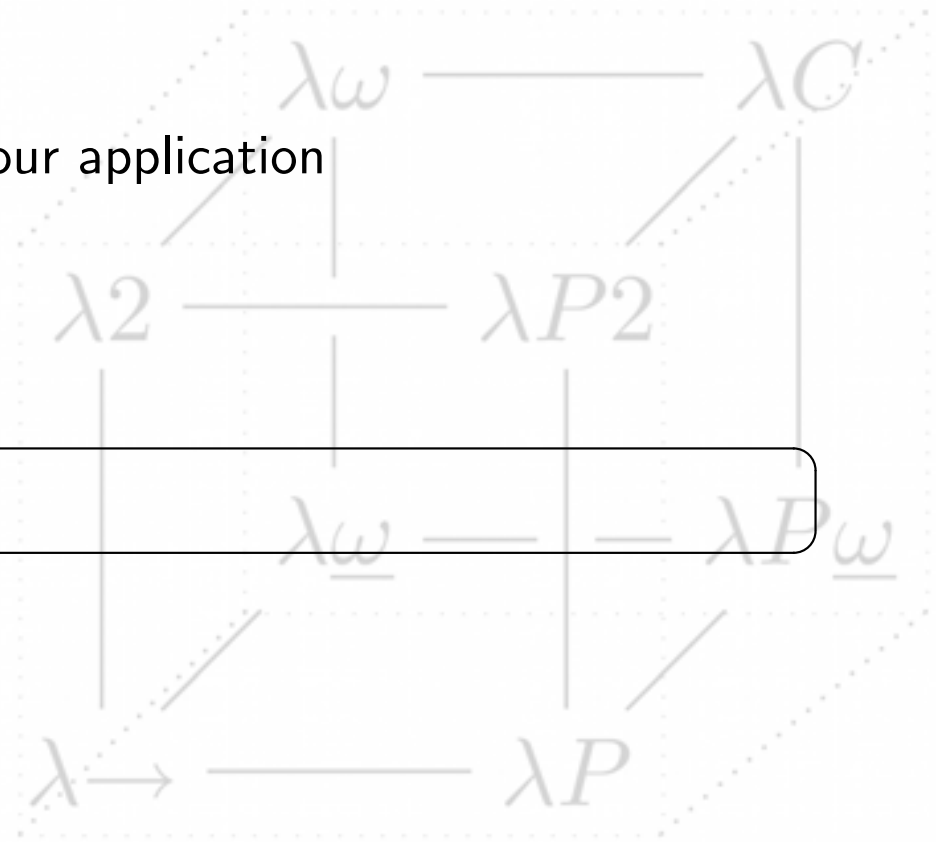
What's inside

```
+ MyApp/Makefile.PL
+ MyApp/t/002_index_route.t
+ MyApp/t/001_base.t
```

- Main module Makefile:
  - usefull to make module of your application
- Your test suite;

```
+ MyApp/lib/MyApp.pm
```

- Your application!



Part 2: Flamenco

Traffic Control

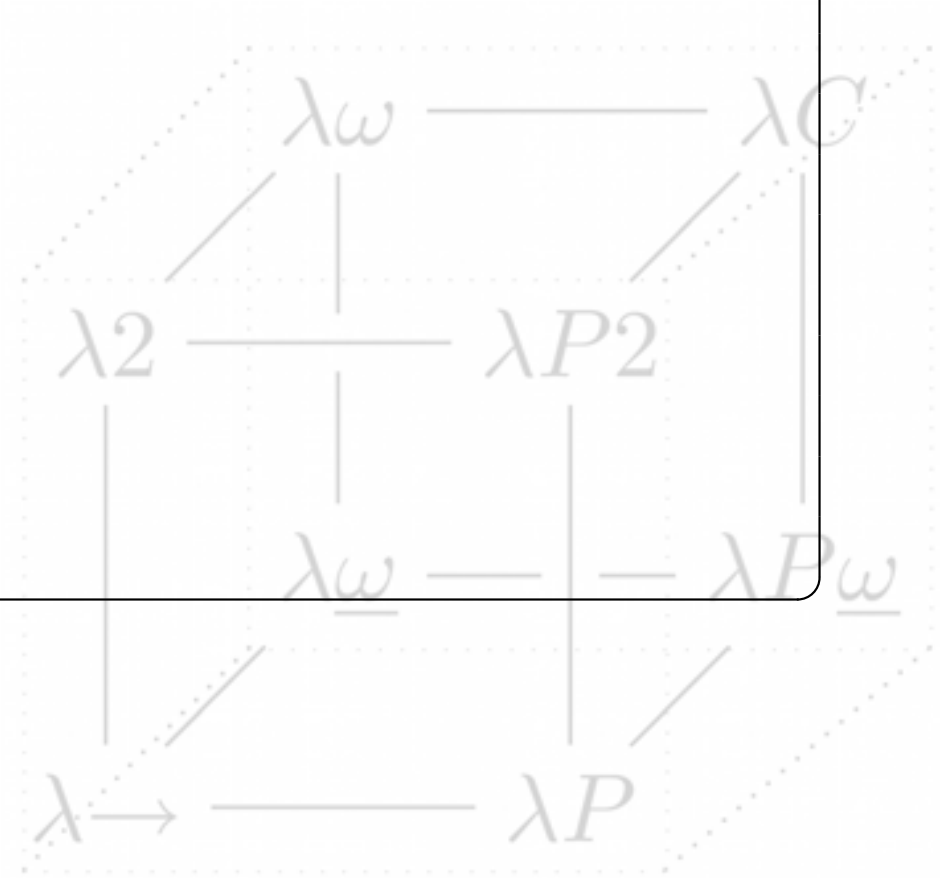
Your `MyApp/lib/MyApp.pm` includes:

```
package MyApp;
use Dancer ':syntax';

our $VERSION = '0.1';

get '/' => sub {
    template 'index';
};

true;
```



How to test it?

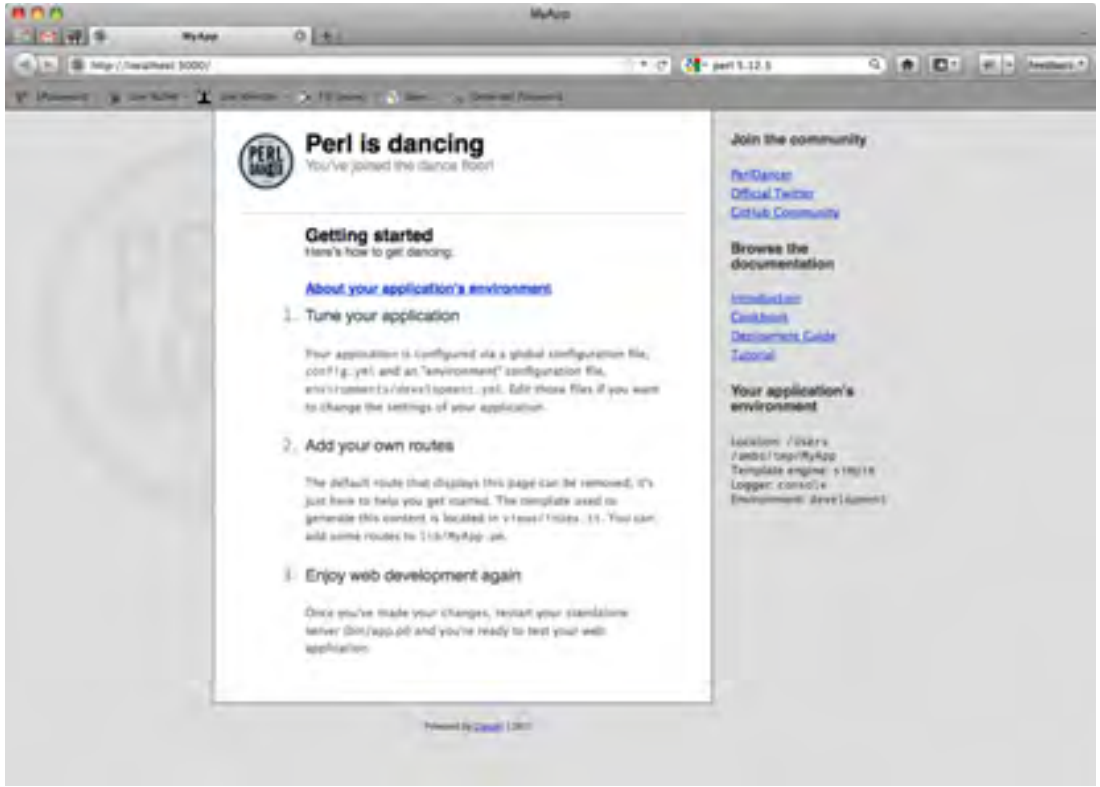
Start the standalone server,

```
[ambs@rachmaninoff MyApp]$ bin/app.pl
```

that shows debug info,

```
[9387] core @0.000017> loading Dancer::Handler::Standalone
handler in /opt/lib/perl5/site_perl/5.12.3/Dancer/Handler.pm l. 39
[9387] core @0.000442> loading handler 'Dancer::Handler::
Standalone' in /opt/local/lib/perl5/site_perl/5.12.3/Dancer.pm l. 230
>> Dancer 1.3011 server 9387 listening on http://0.0.0.0:3000
\subsection{ Entering the development dance floor ...}
```

and open a browser in the specified port.


Perl is dancing
You've joined the dance floor!
Getting started
Here's how to get dancing.
About your application's environment
1. Tune your application
2. Add your own routes
3. Enjoy web development again
Join the community
Browse the documentation
Your application's environment

It Works!

Route Handlers

- A Dancer app is a collection of route handlers;
- A route handler is, basically, a sub;
- It is bound to an http method;
- And to a path or a path pattern;

Example

```
get '/' => sub { ... };

post '/submit/:file' => sub { ... };

del '/resource/*' => sub { ... };
```

Route Handlers

- A Dancer app is a collection of route handlers;
- A route handler is, basically, a sub;
- It is bound to an http method;
- And to a path or a path pattern;

Example

```
get '/' => sub { ... };

post '/submit/:file' => sub { ... };

del '/resource/*' => sub { ... };
```

Route Handlers

- Static patterns (paths):

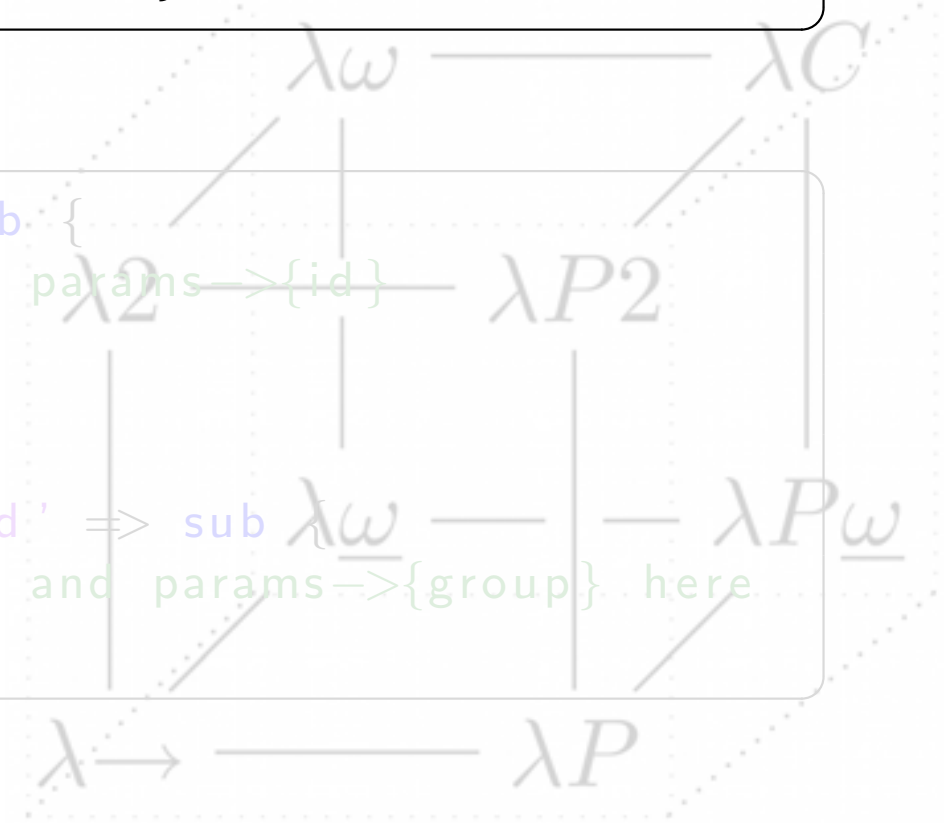
```
get '/' => sub { ... };

get '/about' => sub { ... };
```

- Patterns with named tokens:

```
get '/book/:id' => sub {
  # do something with params->{id}
  ...
};

get '/user/:group/:uid' => sub {
  # use params->{uid} and params->{group} here
};
```



Route Handlers

- Static patterns (paths):

```
get '/' => sub { ... };

get '/about' => sub { ... };
```

- Patterns with named tokens:

```
get '/book/:id' => sub {
  # do something with params->{id}
   ...
};

get '/user/:group/:uid' => sub {
  # use params->{uid} and params->{group} here
};
```

Route Handlers

- Patterns with anonymous tokens:

```
get '/file/*.*' => sub {
  my ($file, $ext) = splat;
  ...
}

get '/show/*/*' => sub {
   my ($cat, $subcat) = splat;
};
```

- Regular expressions:

```
get qr{post/(\d+)-(\d+)-(\d+)} => sub {
   my ($year, $month, $day) = splat;
}
```

Route Handlers

- Patterns with anonymous tokens:

```
get '/file/*.*' => sub {
  my ($file, $ext) = splat;
  ...
}

get '/show/*/*' => sub {
   my ($cat, $subcat) = splat;
};
```

- Regular expressions:

```
get qr{post/(\d+)-(\d+)-(\d+)} => sub {
   my ($year, $month, $day) = splat;
}
```

Part 3: Mambo

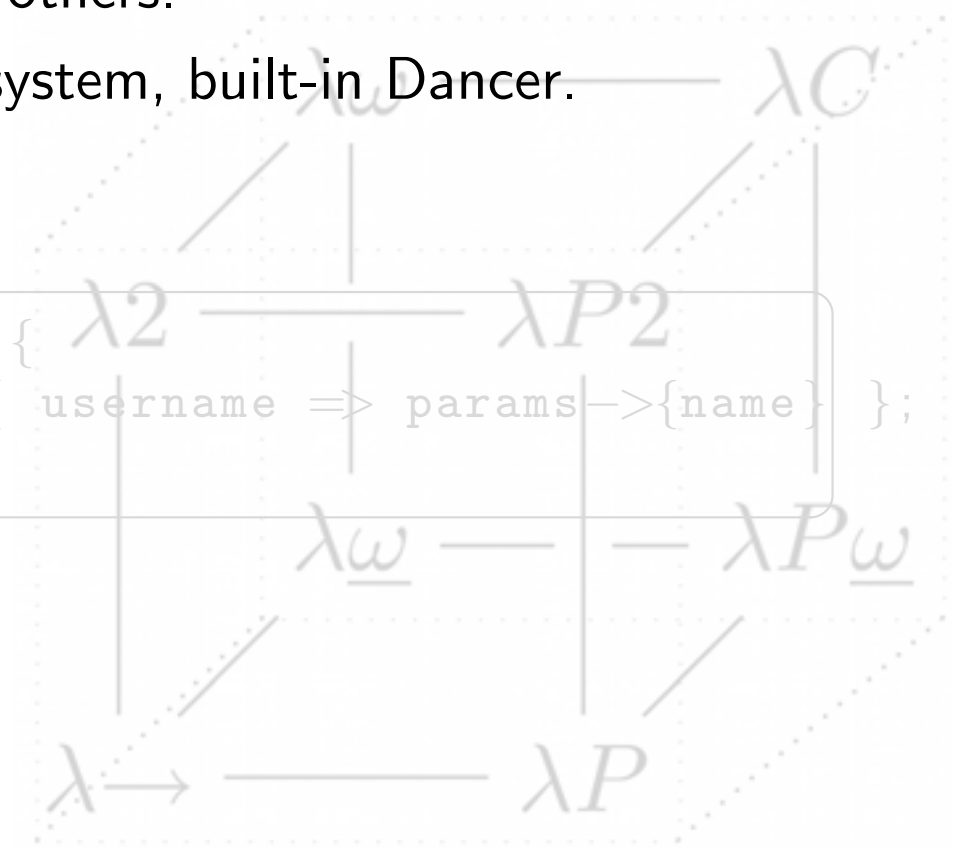
Templating

Dancer has plugins for most templating systems:

- Mason, Template Toolkit and others.
- Default is a Simple template system, built-in Dancer.

Use the template inside the route:

```
get '/user/:name' => sub {
  template 'profile' => { username => params->{name} };
};
```



Templating

Dancer has plugins for most templating systems:

- Mason, Template Toolkit and others.
- Default is a Simple template system, built-in Dancer.

Use the template inside the route:

```
    get '/user/:name' => sub {
      template 'profile' => { username => params->{name} };
    };
```

Rerouting

You can reroute by:

Passing the control to the next matching handler:

```
get '/lazy' => sub {
  pass and return false;
};
```

Redirecting to other URI:

```
get '/forbidden' => sub {
  return redirect '/better/place'
};
```

Rerouting

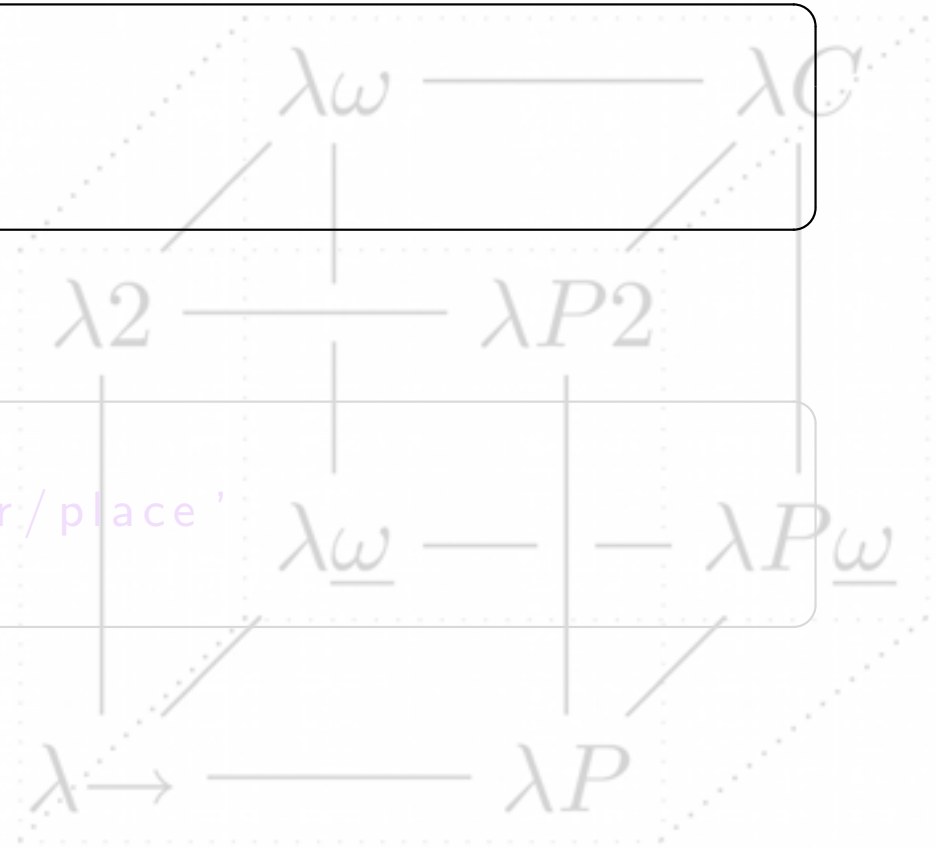
You can reroute by:

Passing the control to the next matching handler:

```
get '/lazy' => sub {
  pass and return false;
};
```

Redirecting to other URI:

```
get '/forbidden' => sub {
  return redirect '/better/place'
};
```



Rerouting

You can reroute by:

Passing the control to the next matching handler:

```
get '/lazy' => sub {
  pass and return false;
};
```

Redirecting to other URI:

```
get '/forbidden' => sub {
  return redirect '/better/place'
};
```

Serving Files

You can serve a static file:

```
get '/dowload/:file' => sub {
  my $file = params->{file};

  pass and return false unless -f $file;

  send_file $file;
};
```

If the content is generated, just change content-type:

```
get '/readme.txt' => sub {
  content_type 'text/plain';
  return 'this is plain text'
};
```

Serving Files

You can serve a static file:

```
get '/dowload/:file' => sub {
  my $file = params->{file};

  pass and return false unless -f $file;

  send_file $file;
};
```

If the content is generated, just change content-type:

```
get '/readme.txt' => sub {
  content_type 'text/plain';
  return 'this is plain text'
};
```

Thanks to Alexis Sukrieh

(I stole some slides)