

Dancer Academy - From Zero to Hero

Stefan Hornburg (Racke)

`racke@linuxia.de`

Perl Mongers, Ljubljana, 8th June 2013

American Spaces Applications

- ▶ Dashboard <https://americanspaces.state.gov>
- ▶ eLibraryUSA <http://elibraryusa.state.gov>
- ▶ eShop <https://eshop.state.gov>
- ▶ LDAP administration

Easy to start with

- ▶ Application ready to go
- ▶ Syntax easy to understand
- ▶ Routes and Keywords

Easy to expand

- ▶ Plugins
- ▶ Hooks
- ▶ Engines

Quickstart I

- ▶ cpanm Dancer
- ▶ cpanm YAML
- ▶ dancer -a Dropbox

Program

```
./bin/app.pl
```

```
#!/usr/bin/env perl  
use Dancer;  
use Dropbox;  
dance;
```

Module

```
lib/Dropbox.pm
```

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

```
our $VERSION = '0.1';
```

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

```
true;
```

Configuration Files

`config.yml`

`environments/development.yml`

`environments/production.yml`

Static Files

css / style .css

css / error .css

images / perldancer-bg .jpg

images / perldancer .jpg

javascripts / jquery .js

500.html

404.html

favicon .ico

dispatch .cgi

Template

views/**index**.tt

views/layouts/main.tt

Quickstart II

- ▶ `cd Dropbox`
- ▶ `./bin/app.pl`
- ▶ `x-www-browser http://localhost:3000/`

Templates

Layout

`views/layouts/main.tt`

Content

`views/index.tt`

Templates

- ▶ Normal Layout

```
template 'index', {name => 'Test'}
```

- ▶ Specific Layout

```
template 'index', {name => 'Test'}, {layout => 'test'}
```

- ▶ No Layout

```
template 'index', {name => 'Test'}, {layout => undef}
```

Template for Filebrowser

```
<h1><% directory %></h1>
<table>
<tr><th>Name</th><th>Type</th><th>Modified</th></tr>
<% FOREACH file IN files %>
<tr><td><% file.name %></td>
    <td><% file.type %></td>
    <td><% file.modified %></td></tr>
<% END %>
</table>
```

Route for Filebrowser

```
get '/home' => sub {  
  my $files = autoindex('/');  
  
  template 'filebrowser', {directory => 'Home',  
                           files => $files ,  
                           };  
};
```

Configuration

Standard config.yml:

```
# template engine  
# simple: default and very basic template engine  
template: "simple"
```

Dropbox config.yml:

```
template: "template_toolkit"
```


Routes and Keywords

- ▶ HTTP method
 - ▶ get
 - ▶ post
 - ▶ ...
 - ▶ any
- ▶ Path
- ▶ Subroutine

Routes

- ▶ String
- ▶ Named parameters
- ▶ Wildcards
 - ▶ Splat
 - ▶ Megasplat
- ▶ Regular expression
- ▶ Regular expression with captures

String

```
get '/home' => sub {  
  my $files = autoindex('/');  
  
  template 'filebrowser', {directory => 'Home',  
                           files => $files ,  
                           };  
};
```

Named parameters

```
get '/home/:file' => sub {  
  my $files = autoindex(param('file'));  
  
  template 'filebrowser', {directory => param('file'),  
                           files => $files,  
                           };  
};
```

Splat

```
get '/images/covers/*.jpg' => sub {  
  my ($isbn) = splat;  
  
  if (-f "public/images/covers/$isbn.jpg") {  
    return send_file "images/covers/$isbn.jpg";  
  }  
  
  status 'not_found';  
  forward 404;  
}
```

Megasplat

`https://eshop.state.gov/lostpwd/biz@linuxia.de/e642bd543b9907bd2c06a`

```
get '/lostpwd/**' => sub {  
  my ($email, $hash) = splat;  
  
  form->fill(email => $email,  
             hash => $hash);  
  
  template('lostpwd_confirm', form => $form);  
}
```

Regular Expression

Catch-All (last route!)

```
any qr{.*} => sub {  
  ...  
};
```

Regular Expression with Captures

`https :// dropbox . nite . si /~ racke / talks / dancer-beamer . pdf`

```
any qr{^/~(?<user>[^/]+)/(?<file>.*?)/?$} => sub {  
  my ($caps, $user, $file);
```

```
  $caps = captures;  
  $file = $caps->{file};  
  $user = $caps->{user};
```

```
  ...
```

```
};
```


Keywords

- ▶ get, post, any, put, del, ...
- ▶ request, params, param
- ▶ redirect, forward, status, header
- ▶ config, var, session
- ▶ from_json, to_json, from_xml, to_xml

var(s) and session

Storing and retrieving data for the current request:

```
var bar => 'pivo';  
$bar = var 'bar';  
$bar = vars->{bar};
```

Storing and retrieving data from the session:

```
session username => 'racke@linuxia.de';  
  
if (! session('username')) {  
    redirect uri_for('/login');  
}
```

Customizing your application

- ▶ Engines
- ▶ Hooks
- ▶ Plugins
- ▶ Middleware

Engines

- ▶ Templates
- ▶ Sessions
- ▶ Logs

Template Engines

- ▶ `Dancer::Template::Toolkit`
- ▶ `Dancer::Template::Xslate`
- ▶ `Dancer::Template::TemplateFlute`
- ▶ ... dozens more

Configuration

```
template: "template_flute"

engines:
  template_flute:
    filters:
      date:
        options:
          format: "%m/%d/%Y"
          strict:
            empty: 0
```

Session Engines

`Dancer::Session::YAML` File based, development only

`Dancer::Session::Storable` File based

`Dancer::Session::DBI`

`Dancer::Session::Memcached`

`Dancer::Session::MongoDB`

`Dancer::Session::Cookie` Session data stored encrypted in a cookie.

Sessions: Storable

- ▶ Engine

`session: Storable`

- ▶ Directory

`session_dir: /var/run/dancer-sessions`

- ▶ Expiration

`session_expires: 8 hours`

Sessions: Cookie

- ▶ Engine

`session: Cookie`

- ▶ Key

`session_cookie_key: kjsdf07234hjfdskf1j12*&(@*j`

Logger Engines

Dancer::Logger::Console

Dancer::Logger::File File in logs/ directory, e.g.
logs/production.log

Dancer::Logger::Syslog

Dancer::Logger::Log4perl

Logger Configuration

- ▶ Engine

`logger: file`

- ▶ Directory

`log_path: log`

- ▶ File

`log_file: error.log`

- ▶ Format

`logger_format: "%t [%P] %L @%D> %m in %f l. %l"`

Hooks

- ▶ What is a hook?
- ▶ `before hook`
- ▶ `before_template_render hook`

What is a Hook

- ▶ Events inside core and plugins
- ▶ Hook into the path of execution
- ▶ Subroutines in your application
- ▶ Multiple hook routines

before Hook

Password protected site:

```
hook 'before' => sub {  
  unless (session('user'))  
    || request->path eq '/login'  
    || request->path =~ m%^/lostpwd%  
  ) {  
    redirect '/login';  
  }  
};
```

Files in public aren't protected!

before_template_render Hook

- ▶ `template 'index', {message => 'Hello'}`
- ▶ add default tokens (params, vars, session)
- ▶ `before_template_render` hook
- ▶ rendering
- ▶ `after_template_render` hook

before_template_render Hook

```
hook before_template_render => sub {  
  my $tokens = shift;  
  my $menu;  
  
  if (session('user')) {  
    $menu = [{name => 'Logout', url => '/logout'}];  
  }  
  else {  
    $menu = [{name => 'Login', url => '/login'}];  
  }  
  
  $tokens->{menu} = $menu;  
};
```


Plugins

- ▶ Features
- ▶ Keywords
- ▶ Routes
- ▶ Configuration

Plugins on CPAN

- ▶ Count: more than 100

- ▶ Overview

`https://metacpan.org/module/Dancer::Plugins`

- ▶ Search

`https://metacpan.org/search?q=dancer+plugin`

Plugins on CPAN

- ▶ `Dancer::Plugin::Database`
database
- ▶ `Dancer::Plugin::Email`
email
- ▶ `Dancer::Plugin::Auth::Extensible`
`require_login, require_role, ...`
`/login, /login-denied/`

Dancer::Plugin::Database Example

```
my $user = database->quick_select( 'users ',  
    {username => params->{'username'}});  
  
if ($user) {  
    session user => params->{'username'};  
}
```

Dancer::Plugin::Database Configuration

```
plugins :  
  Database :  
    driver : Pg  
    database : dropbox  
    username : vienna  
    password : nevairbe
```

More SQL Plugins

- ▶ `Dancer::Plugin::DBIC`
- ▶ `Dancer::Plugin::SimpleCRUD`
- ▶ ...

Dancer::Plugin::Email Example

```
$message = template( 'lostpwd_email',  
    {password => $password, url => $url},  
    {layout => undef});  
  
email ({type => 'html',  
    from => 'service@linuxia.de',  
    to => param( 'email' ),  
    subject => 'Forgot password',  
    message => $message});
```

Dancer::Plugin::Email Redirect

- ▶ Module:

`Email::Sender::Transport::Redirect`

- ▶ Configuration:

```
plugins :
```

```
  Email :
```

```
    transport :
```

```
      Sendmail :
```

```
        redirect_address : biz@icdev.de
```


Dancer::Plugin::Auth::Extensible require_login

```
get '/home/**' => require_login sub {  
  ...  
};
```

Dancer::Plugin::Auth::Extensible require_role

```
get '/accounts' => require_role Administrator sub {  
  template 'accounts', ...  
};
```

Dancer::Plugin::Auth::Extensible Providers

- ▶ Config
- ▶ Unix
- ▶ Database
- ▶ LDAP
- ▶ Subclass

`D::P::Auth::Extensible::Provider::Base`

Write Your Own Plugin

```
package Dancer::Plugin::SSO;  
  
use Dancer::Plugin;  
  
register sso_generate_token => sub {  
  ...  
};  
  
register_plugin;
```

Hooks in Your Own Plugin

Plugin:

```
register_hook qw/sso_login sso_failure/;
```

Application:

```
hook sso_login => sub {  
  my $user_ref = shift;  
  
  session user => $user_ref->username;  
};
```

Middleware Example

```
#!/usr/bin/env perl

use Dancer;
use Dropbox;
use Plack::Builder;

my $app = sub {
    my $env = shift;
    my $request = Dancer::Request->new(env => $env);
    Dancer->dance($request);
};
```

Middleware Example

```
builder {  
  enable SizeLimit => (  
    max_unshared_size_in_kb => 102400, # 100 Mb  
    check_every_n_requests => 10,  
  );  
  
  $app;  
};
```

Middleware in Configuration

plack_middlewares :

- - SizeLimit
 - max_unshared_size_in_kb
 - 102400

Download/Upload

- ▶ File download
- ▶ Upload object
- ▶ Upload form + code

File Download

```
get '/home/**' => sub {  
  my ($parts) = splat;  
  my $path_relative = join('/', @$parts);  
  my $path = autoindex_file_path($path_relative);  
  
  if (-f $path) {  
    return send_file $path, system_path => 1;  
  }  
  elsif (-d $path) {  
    return template 'filebrowser', {directory => pop(@$parts),  
                                     files => autoindex($path_relative)};  
  }  
};
```

Upload Object

```
upload( 'upload_file' ) => Dancer::Request::Upload
```

filename Filename sent by user

basename Stripped filename

size Size in bytes

type Content type

content File content

copy_to Copies temporary file

Upload HTML-Form

```
<div class="upload">  
<h2>Upload file </h2>  
<form name="upload" id="upload_form" action="" method="post"  
  enctype="multipart/form-data">  
<input type="file" name="upload_file"><br>  
<input type="submit" value="OK">  
</form>  
</div>
```

Upload Route

```
post '/home/**' => sub {  
  my ($parts) = splat;  
  my $path_relative = join('/', @$parts);  
  my $path = autoindex_file_path($path_relative);  
  
  if ($upload_file = upload('upload_file')) {  
    # upload file to directory  
    my $target_file = join('/', $path, $upload_file->basename);  
  
    unless ($upload_file->copy_to($target_file)) {  
      die 'Upload failed';  
    }  
  }  
  
  redirect "/home/$path_relative";  
}
```

Security Concerns

- ▶ Input sanitization
- ▶ Path traversal
- ▶ XSS
- ▶ Database injection (SQL/LDAP)
- ▶ POST request limits

Path traversal

Example:

`https://dropbox.nite.si/dropbox/racke@linuxia.de/../../../../etc/passwd`

Protection:

```
if (grep $_ eq '..', @path) {  
    return status 403;  
}
```

Error Handling: 500

- ▶ Static File: `public/500.html`
- ▶ Template: `error_template`
- ▶ Stacktrace: `show_errors`

Error Handling: 404

- ▶ Static File: `public/404.html`
- ▶ any `qr{.*}` => **sub** {
 status 'not_found';
 template 'my_404', { path => request->path };
};
- ▶ Prevent XSS attacks

Deployment

- ▶ Scenarios
- ▶ Starman
- ▶ Nginx
- ▶ Scripts

Scenarios

- ▶ Standalone
 `./bin/app.pl`
- ▶ CGI, FastCGI
- ▶ plackup
 - ▶ Starman
 - ▶ Twiggy
 - ▶ Corona
- ▶ Reverse Proxy

Scenarios: Reverse Proxy

- ▶ Server

- ▶ Apache
- ▶ nginx
- ▶ Perlbal

- ▶ Dancer Configuration

`behind_proxy: 1`

plackup and Starman

► Command line

```
plackup -E production  
      -s Starman \  
      --workers=5 \  
      --pid /home/racke/Dropbox/run/Dropbox.pid \  
      -p 5000 \  
      -a bin/app.pl \  
      -D
```

► Init script

```
► Daemon::Control
```

Nginx Configuration

```
server {  
    listen 80;  
    server_name elibraryusa.state.gov;  
    root /home/dancer/Elib/public;  
  
    ...  
}
```

Nginx Configuration

```
server {  
    ...  
  
    location / {  
        try_files $uri @proxy;  
        access_log off;  
        expires max;  
    }  
  
    ...  
}
```

Nginx Configuration

```
server {  
    ...  
  
    location @proxy {  
        proxy_set_header Host $http_host;  
        proxy_set_header X-Forwarded-Host $host;  
        proxy_set_header X-Real-IP $remote_addr;  
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  
        proxy_pass          http://localhost:5001;  
    }  
}
```


Nginx Configuration SSL

```
server {  
    listen 213.129.249.66:443;  
    server_name eshop.state.gov;  
    root /home/interch/eShop/public;  
  
    ssl on;  
    ssl_certificate_key /etc/ssl/private/eshop.state.gov.key;  
    ssl_certificate /etc/ssl/certs/eshop.state.gov.pem;  
  
    ...  
}
```

Nginx Configuration SSL

```
server {  
    ...  
  
    location @proxy {  
        proxy_set_header Host $http_host;  
        proxy_set_header X-Forwarded-Host $host;  
        proxy_set_header X-Real-IP $remote_addr;  
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  
        proxy_set_header X-Forwarded-Proto https;  
        proxy_pass http://localhost:5001;  
    }  
}
```

Scripts

```
use Docker ':script';  
  
set logger => 'console';  
set logger_format => '%m';
```

Dancer Problems

- ▶ Merging configuration files
- ▶ `prefix` is global
- ▶ Session key deletion `session user => undef`

Dancer2

- ▶ Pure OO (Moo)
- ▶ Globals
- ▶ Scoping for Subapplications
- ▶ YAML => Config::Any

Community

Web: <http://perldancer.org/>

Github: [git://github.com/PerlDancer/Dancer.git](https://github.com/PerlDancer/Dancer.git)

[git://github.com/PerlDancer/Dancer2.git](https://github.com/PerlDancer/Dancer2.git)

Dropbox plugin: [https:](https://metacpan.org/module/Dancer::Plugin::Dropbox)

[//metacpan.org/module/Dancer::Plugin::Dropbox](https://metacpan.org/module/Dancer::Plugin::Dropbox)

IRC: #dancer @ irc.perl.org

Contribution: [Dancer::Development](#)

The End

Slides: <http://www.linuxia.de/talks/lj2013/dancer-beamer.pdf>