Tanz!

Stefan Hornburg (Racke)
racke@linuxia.de

13. Deutscher Perl-Workshop, Frankfurt, 21. Oktober 2011

Contents

1 Hintergrund und Projekte

1.1 Tanzflur

Tanzflur

- einfach
- ausdrucksstark
- effektiv
- minimale Abhängigkeiten

Projekte

- Dropbox-Klon
- LDAP Benutzerverwaltung
- Procurement für American Corners

1.2 Dropbox

Für die Firma Caithness in New York habe ich einen Dropbox-Klon entwickelt.
Dropbox demonstriert die automatische Erkennung des MIME-Typs durch Dancer.

Dropbox

- Autoindex
- Upload/Download
- Benutzerverwaltung

1.3 LDAP Benutzerverwaltung

LDAP Benutzerverwaltung

- LDAP-Verzeichnis
- Benutzerverwaltung
  - Adminstrator
  - Referrer
  - Patron

1.4 Procurement für American Spaces

Procurement für American Spaces

- Warenkorb/Wunschliste
- Genehmigungsprozesse
- LibraryThing
- ISBNDB

`https://vsc.state.gov/`

2 Routes, Filter und Hooks

2.1 Routes

Für eine Route benötigen wir

- HTTP-Methode
- Pfad
- Subroutine

Rezept für Routes

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

use Dancer;

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

dance;
```

Der Pfad für eine Route kann in einer der folgenden Weisen angegeben werden.

Routes

- String
- String mit benannten Parametern
- String mit Wildcards
  - Splat
  - Megasplat
- Regulärer Ausdruck
- Regulärer Ausdruck mit Captures

2.1.1 Splat

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;
}
```

2.1.2 Megasplat

Die einfache Wildcard matcht nur auf einen Teil des Pfads, d.h. bis zum nächsten Schrägstrich (Slash).

Mit der doppelten Wildcard (Megasplat) wird einfach der Rest des Pfades gematcht und die `splat`-Funktion gibt eine Liste zurück.

Megasplat

```
https://vsc.state.gov/lostpwd/biz@linuxia.de/e642bd543b9907bd2c06aa485261cb1a849a9f23fc7324bff45ebd3

get '/lostpwd/**' => sub {
    my ($email, $hash) = splat;

    form->fill(email => $email,
               hash => $hash);

    template('lostpwd_confirm', form => $form);
}
```

Captures

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

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

    $capts = captures;
    $file = $capts->{file};
    $user = $capts->{user};

    ...
};
```

2.1.3 Präfix

Präfix

```
prefix '/wiki';

get qr{/?(?<page>.*)} => sub {
    my $capts = captures;

    template 'wiki', {content => wiki($capts->{page}),
                      page => $capts->{page}};
};
```

2.2 Filters

Filters

```
before sub {
};

after sub {
};
```

before

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

2.3 Hooks

Hooks

```
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;
};
```

Die folgenden Hooks existieren in Dancer:

- before_deserializer
- before_file_render

- before_error_init
- before_error_render
- before_template_render
- before_layout_render
- before_serializer
- after_deserializer
- after_file_render
- after_template_render
- after_layout_render
- after_error_render

Der after_file_render-Hook wird ausgelöst, nachdem eine statische Datei (Bild oder CSS-Datei) gesendet wurde.

3 Plugins

Plugins

- Funktionalität
- Schlüsselwort
- Route
- Konfiguration

3.1 Plugins auf CPAN

Plugins auf CPAN

- Dancer::Plugin::Database `database`
- Dancer::Plugin::Email `email`
- Dancer::Plugin::Ajax `ajax`

Dancer::Plugin::LibraryThing Konfiguration

```
plugins:
    LibraryThing:
        api_key: d231aa37c9b4f5d304a60a3d0ad1dad4
        directory: public/images/covers
        size: large
```

Dancer::Plugin::LibraryThing Code

```
use Dancer::Plugin::LibraryThing;

get '/images/covers/*.jpg' => sub {
    my ($isbn, @ret);
    $isbn = splat;

    unless (-f "public/images/covers/$isbn.jpg") {
        @ret = librarything_cover($isbn);
        if (@ret < 3) {
            status 'not_found';
            forward 404;
        }
    }

    return send_file "images/covers/$isbn.jpg";
}
```

3.2 Writing Plugins

Writing Plugins

- register
- register_plugin
- plugin_setting

3.3 Plugins und Hooks

Plugins und Hooks

```
Dancer::Factory::Hook
        ->instance
        ->install_hooks('before_cart_add');
```

4 Deployment

4.1 Sessions

Die Session-Engine wird mit `session` konfiguriert, ansonsten werden keine Sessions verwendet. Cookies sind Voraussetzung für Sessions mit Dancer.

YAML-Sessions und JSON-Sessions sind nur für die Entwicklung gedacht

Alternativen sind u.a.:

- Storable
- Cookie
- Memcached

- MongoDB

Eine gute Idee ist es die Session Engine für den Produktionsbetrieb in `config.yml` zu konfigurieren und für die Entwicklung in `environments/development.yml` zu überschreiben.

Im Produktionsbetrieb ist es empfehlenswert die Ablauffrist für Sessions zu setzen, sonst ist eine Session für immer gültig.

Sessions

- Engine `session: Storable`
- Verzeichnis `session_dir: /var/run/dancer-sessions`
- Ablauffrist `session_expires: 8 hours`

4.2 Logs

Die Logger-Engine wird mit `logger` konfiguriert, ansonsten werden Logs erzeugt.

Das Format für die Logs wird mit `logger_format` definiert, siehe `http://search.cpan.org/perldoc?Dancer::Logger::Abstract#logger_format`.

Verfügbare Engines sind u.a.:

- console
- file
- syslog *
- log4perl *

Die Rotation der Logs für dateibasierte Engines darf nicht vergessen werden im Produktionsbetrieb.

Logs

- Engine `logger: file`
- Verzeichnis `log_path: log`
- Format `logger_format: "%t [%P] %L @%D> %m in %f l. %l"`

4.3 Szenarien

Die möglichen Szenarien für ein Deployment von Dancer findet man in der Dancer-Dokumentation: `http://search.cpan.org/perldoc?Dancer::Deployment`.

Szenarien

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

Szenarien: Reverse Proxy

- Apache
- nginx
- Perlbal

4.4 Starman

Starman

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

4.5 Perlbal

Am Anfang der Konfigurationsdatei laden wir das “vpaths” Plugin.

Dann binden wir für bessere Performache `http://search.cpan.org/perldoc?Perlbal::XS::HTTPHe` ein.

Perlbal

```
LOAD vpaths
XS enable headers
```

4.5.1 Pools

Für unsere beiden Projekte erzeugen wir zunächst Pools:

Perlbal: Pools

```
CREATE POOL prod_starman_dosqua
 POOL prod_starman_dosqua ADD 127.0.0.1:5000

CREATE POOL prod_starman_vsc
 POOL prod_starman_vsc ADD 127.0.0.1:5001
```

Perlbal: Reverse Proxy

```
CREATE SERVICE vsc_prod
 SET role                 = reverse_proxy
 SET pool                 = prod_starman_vsc
 SET buffer_uploads       = on
ENABLE vsc_prod
```

Perlbal: Selector

```
CREATE SERVICE vsc_selector
 SET listen               = 86.59.13.238:80
 SET role                 = selector
 SET plugins              = vpaths
 VPATH .*                 = vsc_prod
ENABLE vsc_selector
```

Perlbal: SSL Selector

```
CREATE SERVICE vsc_ssl_selector
 SET listen               = 86.59.13.238:443
 SET role                 = selector
 SET plugins              = vpaths
 SET enable_ssl           = on
 SET ssl_key_file         = /etc/ssl/private/vsc.state.gov.key
 SET ssl_cert_file        = /etc/ssl/certs/vsc.state.gov.pem
 VPATH .*                 = vsc_prod
ENABLE vsc_ssl_selector
HEADER vsc_ssl_selector INSERT X-Forwarded-Proto: HTTPS
```

4.6 Skripts

Dancer-Skripts sollten innerhalb der Dancer-Applikation liegen, gut geeignet ist das bin/-Verzeichnis. Ein symbolischer Link is ausreichend. Ansonsten wird die Konfiguration nicht gefunden.

Eine Ausgabe von Meldungen auf der Konsole ist wünschenswert, mit `%m` wird lediglich die reine Meldung ausgegeben.

Skripts

```
use Dancer ':script';

set logger => 'console';
set logger_format => '%m';
```

5 Ausblick

5.1 Dancer 2

Dancer2

- keine globalen Variablen
- 100% OO Backend (Moo)
- Scoping for Sub-Applikationen
- überarbeitete Architektur

5.2 Community

Community

Web: `http://perldancer.org/`
Github: `git://github.com/sukria/Dancer.git`
IRC: #dancer @ irc.perl.org
Mitarbeit: Dancer::Development

The End

Slides, Handout, Skripte: `http://www.linuxia.de/talks/pws2011/`