Introduction
Dancer::Plugin::DBIC
Dancer::Session::DBIC
TableEditor
Forecast and Contribution

Dancer and DBIx::Class

Stefan Hornburg (Racke) racke@linuxia.de

Czech Perl Workshop, Prague, 21st May 2014



Easy to start with

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

Easy to start with

- cpanm Dancer YAML
- dancer -a Dropbox
- cd Dropbox
- ./bin/app.pl

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;
```

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}
```

Forecast and Contribution

Routes and Keywords

- HTTP method
 - get
 - post
 - ٠...
 - any
- Path
- Subroutine

Routes

- String
- Named parameters
- Wildcards
 - Splat
 - Megasplat
- Regular expression



String

Named parameters

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

Regular Expression

```
Catch-All (last route!)
any qr{.*} => sub {
    ...
};
```

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');
}
```

Easy to expand

- Plugins
- Hooks
- Engines

before Hook

```
Password protected site:
```

Solid

- Stable
- Keep behaviour
- Community

Applications

```
► Simple Dropbox https:
//metacpan.org/pod/Dancer::Plugin::Dropbox
```

```
.state.gov Websites
https://eshop.state.gov/
```

Monitor for Power Plant

DBIx::Class

- ► ORM
- Objects instead of SQL
- Performance

DBIx::Class

- Database => Schema
- interchange6 => Interchange6::Schema
- Table => Result classes
- users => Interchange6::Schema::Result::User
- Queries => Result sets



User and Roles

User ▶ racke@linuxia.de

info@nite.si

▶ test@linuxia.at

Role

user

editor

admin

guest

Tables

- users
 - users id
 - email
 - first_name
 - **>**
- roles
 - roles id
 - name
 - label
- user roles
 - users id
 - roles_id



Roles for an user

```
mysql> select R.name from users U
    join user_roles UR on (U.users_id = UR.users_id)
    join roles R on (UR.roles_id = R.roles_id)
    where U.email = 'racke@linuxia.de';
```

User with DBIx::Class

```
$rs = $schema->resultset('User');
$user = $rs->find(1);
$user = $rs->find({email => 'racke@linuxia.de'});
$first_name = $user->first_name;
$users_linuxia = $rs->search({
    email => {like => '%@linuxia.de'}});
```

Roles with DBIx::Class

```
$rs = $schema->resultset('User');
$user = $rs->find({email => 'racke@linuxia.de'});
$roles = $user->roles;
```

User Result Class

```
package Interchange6::Schema::Result::User;

__PACKAGE__->table("users");

__PACKAGE__->add_columns(...);

PACKAGE _->set primary key("users id");
```

User Result Class

```
package Interchange6::Schema::Result::User;

__PACKAGE__->has_many("UserRole",
    "Interchange6::Schema::Result::UserRole",
    { "foreign.users_id" => "self.users_id" },
);

__PACKAGE__->many_to_many("roles", "UserRole", "Role");
```

Object inflation

Object vs Hashref

- Debug / Logs
- Templates
- ► API / JSON
- Speed

Forecast and Contribution

Dancer DBIx::Class Database Administration

Database Administration

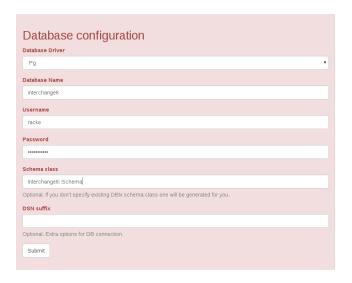
- phpmyadmin
- phppgadmin
- TableEditor

TableEditor Features

- Different database systems MySQL, PostgreSQL, ...
- higher level of abstraction
- modern frontend
- concise source code
- "simple" installation



Input Database Parameters



View Products

Configuration Status

Product - List of items



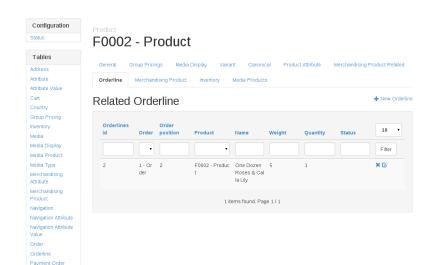


View Product

Weight

Configuration Status F0002 - Product Tables General Canonical Merchandising Product Related Merchandising Product **Attribute** Media Products Group Pricings Media Display Variant Sku Cart F0002 Name One Dozen Roses & Calla Lilv Media Display Short description Media Product What says I love you better than 1 dozen fresh roses with calla lily? Media Type Merchandising Description Surprise the one who makes you smile, or express yourself perfectly with this stunning bouquet of one dozen fresh red roses. This elegant Merchandising Navigation Price Navigation Attribute 49.95 Navigation Attribute Value Uri Order one-dozen-red-roses-calla-lilly Orderline Payment Order

Relationship Orderline



Overview Dancer::Plugin::DBIC

- Usage
- Configuration
- ► UTF-8
- Create schema dynamically

DBIx::Class without Dancer Plugin

```
use Interchange6::Schema;

$schema = Interchange6::Schema->connect(...);

$schema->resultset('User')->search({..});
```

DBIx::Class with Dancer Plugin

```
use Dancer:: Plugin :: DBIC;
schema->resultset('User')->search({..});
resultset('User')->search({..});
rset('User')->search({..});
```

Usage Configuration UTF-8 Create schema dynamically

Configuration

```
plugins:
   DBIC:
    default:
        dsn: dbi:mysql:interchange6
        user: racke
        pass: nevairbe
        schema_class: Interchange6::Schema
```

Multiple Schemas

```
plugins:
  DBIC:
    default:
      dsn: dbi:mysql:interchange6
      user: racke
      pass: nevairbe
      schema class: Interchange6::Schema
    legacy:
      dsn: dbi:mysql:interchange5
      user: racke
      pass: nevairbe
      schema_class: Interchange5::Schema
```

Multiple Schemas

```
use Dancer::Plugin::DBIC;
schema('legacy')->resultset('UserDb')->search({..});
```

UTF-8 for MySQL

```
plugins:
   DBIC:
    default:
        dsn: dbi:mysql:interchange6
        user: racke
        pass: nevairbe
        schema_class: Interchange6::Schema
        options:
        mysql_enable_utf8: 1
```

Create schema dynamically

- schema_class missing in configuration
- DBIx::Class::Schema::Loader
- test and development
- TableEditor

Overview Dancer::Session::DBIC

- engines
- configuration
- serialization
- session expiration

Engines

- Templates TT, Xslate, Flute, ...
- Sessions
 Storable, Database, DBIC
- LoggerFile, Syslog
- Serializer JSON, YAML, XML



session name of session engine (DBIC) session_options options session expires expiration date

```
session: "DBIC"
session_options:
  dsn: dbi:mysql:interchange6
  user: racke
```

schema_class: Interchange6::Schema

resultset: Session
id_column: sessions_id
data_column: session_data

session_expires: 12 hours

pass: nevairbe

```
set session => 'DBIC';
set session_options => {schema => schema};
```

Example table

```
CREATE TABLE 'sessions' (
    'sessions_id' varchar(255) NOT NULL,
    'session_data' text NOT NULL,
    'created' datetime NOT NULL,
    'last_modified' datetime NOT NULL,
    PRIMARY KEY ('sessions_id')
) ENGINE=InnoDB;
```

Serializer

```
set 'session_options' => {
    schema => schema,
    serializer => sub { YAML::Dump(@_); },
    deserializer => sub { YAML::Load(@_); },
};
```

Session expiration

- remove old sessions from database
- ▶ Interchange6::Schema::Resultset::Session

```
$schema->resultset('Session')->expire('12 hours');
```

Overview TableEditor

- Installation
- Frontend
- Routes
- Login
- Relationships
- Configuration



Introduction
Dancer::Plugin::DBIC
Dancer::Session::DBIC
TableEditor
Forecast and Contribution

Installation Frontend Routes Login Relationships Configuration

Installation

```
git clone https://github.com/interchange/TableEditor
cd TableEditor
cpanm .
./bin/app.pl
```

Introduction
Dancer::Plugin::DBIC
Dancer::Session::DBIC
TableEditor
Forecast and Contribution

Installation Frontend Routes Login Relationships Configuration

Driver

- DBD::mysql
- DBD::Pg
- **...**

Frontend

- Angular
 - Routes for the frontend
 - XHR requests to REST API
 - ► JSON
- Bootstrap
- Theme

Routes

```
get '/:class/:id' => require_login sub {
    # retrieve database record and add relationships
    ...
    return to_json($data, {allow_unknown => 1});
};
```

Login

- Dancer::Plugin::Auth::Extensible
- Provider
 - Unix
 - DBIC
- Database (planned)



Relationships

- belongs to
- has_many
- might_have
- has_one
- many_to_many needs to be configured

- Auth::Extensible
- DBIC
 - ▶ default

Installation Frontend Routes Login Relationships Configuration

Planned Features

- Search (Solr)
- Select schema
- Debian packages

Development

https://github.com/interchange/TableEditor

Dancer2

Slides

Slides: http://www.linuxia.de/talks/czpw2014/dancer-dbic-en-beamer.pdf