Dancer and DBIx::Class

Stefan Hornburg (Racke)
racke@linuxia.de

YAPC::Europe 2014, Sofia, 23rd August 2014

Dancer and DBIx::Class

Stefan Hornburg (Racke)
`racke@linuxia.de`

YAPC::Europe 2014, Sofia, 23rd August 2014

Let's dance. I'm Racke from Hannover.pm in Germany and work as self employeed programmer and system administrator.

Most part of my work consists of eCommmerce projects, which means that amongst other things I'm writing web applications which are using databases.

My presentation today is about Dancer and DBIX::Class and how they play together.

Dancer is a micro web framework which makes it really easy to write your own web application. Really so easy that it even programmers from other languages like PHP and Ruby start with Perl just because of Dancer.

Many web applications are using a relational database like MySQL or Postgres. DBIx::Class is an Object Relational Mapper that provides you with objects instead of just data. This is easier, more convenient and even faster for reading and manipulating your database records.

Introduction

- Dancer
- DBIx::Class
- TableEditor

My presentation starts with a short introduction into Dancer and DBIx::Class. After that I'll show you a general purpose application called TableEditor before we dip into the gory details about how you make Dancer and DBIx::Class working together.
So, let's start to dance first.

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`

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

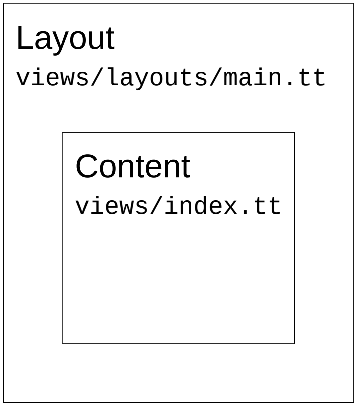
Module

lib/Dropbox.pm

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

our $VERSION = '0.1';

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

true;
```

The content will be rendered first and passed to the layout renderer, so before_layout_render could mangle with it.
The values passed to the template keyword are used for both layout and content.

Templates



Templates

Templates

- Normal Layout

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

- Specific Layout

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

- No Layout

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

Für eine Route benötigen wir

- HTTP-Methode
- Pfad
- Subroutine

Routes and Keywords

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

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

Routes

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

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;
}
```

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://eshop.state.gov/lostpwd/biz@linuxia.de/e642bd543

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 {
    ...
};
```

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 Database, Email, Social Networks
- Hooks
- Engines

before Hook

Password protected site:

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

Solid

- Stable
- Keep behaviour
- Community

DBIx::Class

- ORM
- Objects instead of SQL
- Performance

It took some time to get involved with DBIx::Class for various reasons.

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';

+--------+
| name   |
+--------+
| user   |
| editor |
+--------+
```

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

```
debug "Role: ", $role;

debug "Role: ", {$role->get_inflated_columns};

Role: {'label' => 'User',
       'name' => 'user',
       'roles_id' => '3'}
```

Object vs Hashref

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

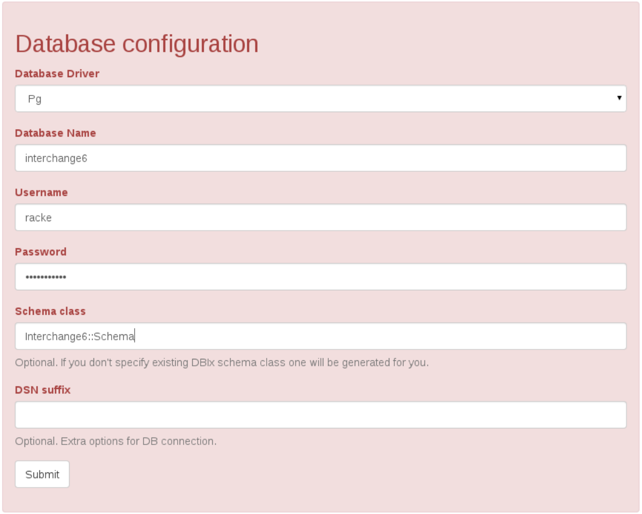
Database Administration

- ~~phpmyadmin~~
- ~~phppgadmin~~
- TableEditor

TableEditor Features

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

Input Database Parameters



Database configuration

Database Driver

Pg

Database Name

interchange6

Username

racke

Password

••••••••••••

Schema class

Interchange6::Schema

Optional. If you don't specify existing DBIx schema class one will be generated for you.

DSN suffix

Optional. Extra options for DB connection.

Submit

View Employees

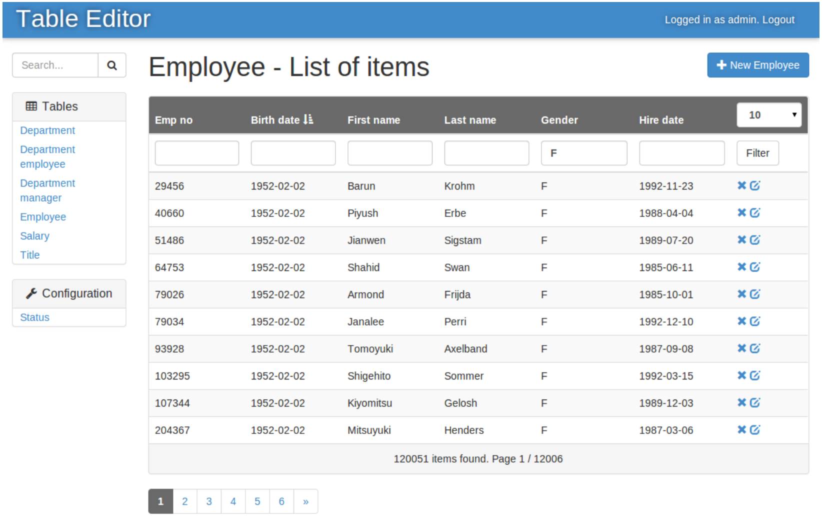


Table Editor

Logged in as admin. Logout

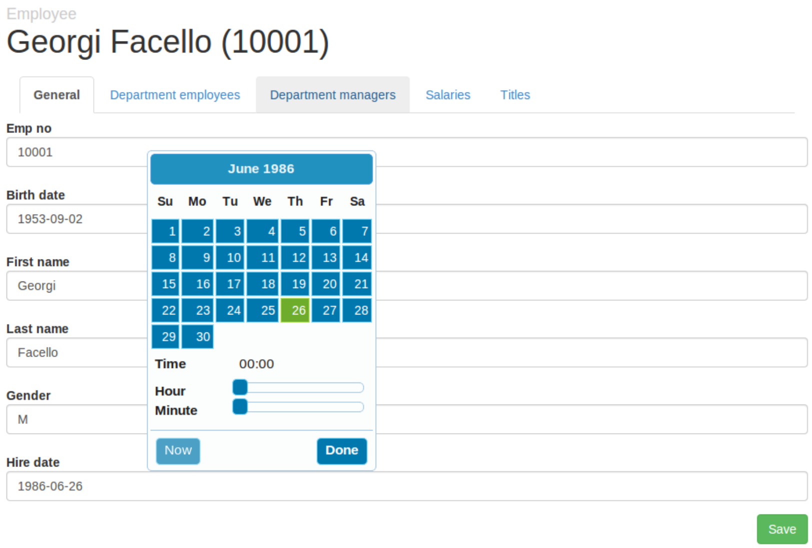
Employee - List of items

Emp no	Birth date	First name	Last name	Gender	Hire date
				F	
29456	1952-02-02	Barun	Krohm	F	1992-11-23
40660	1952-02-02	Piyush	Erbe	F	1988-04-04
51486	1952-02-02	Jianwen	Sigstam	F	1989-07-20
64753	1952-02-02	Shahid	Swan	F	1985-06-11
79026	1952-02-02	Armond	Frijda	F	1985-10-01
79034	1952-02-02	Janalee	Perri	F	1992-12-10
93928	1952-02-02	Tomoyuki	Axelband	F	1987-09-08
103295	1952-02-02	Shigehito	Sommer	F	1992-03-15
107344	1952-02-02	Kiyomitsu	Gelosh	F	1989-12-03
204367	1952-02-02	Mitsuyuki	Henders	F	1987-03-06

120051 items found. Page 1 / 12006

1 2 3 4 5 6 »

Edit Employee



Employee

Georgi Facello (10001)

Emp no	10001
Birth date	1953-09-02
First name	Georgi
Last name	Facello
Gender	M
Hire date	1986-06-26

June 1986

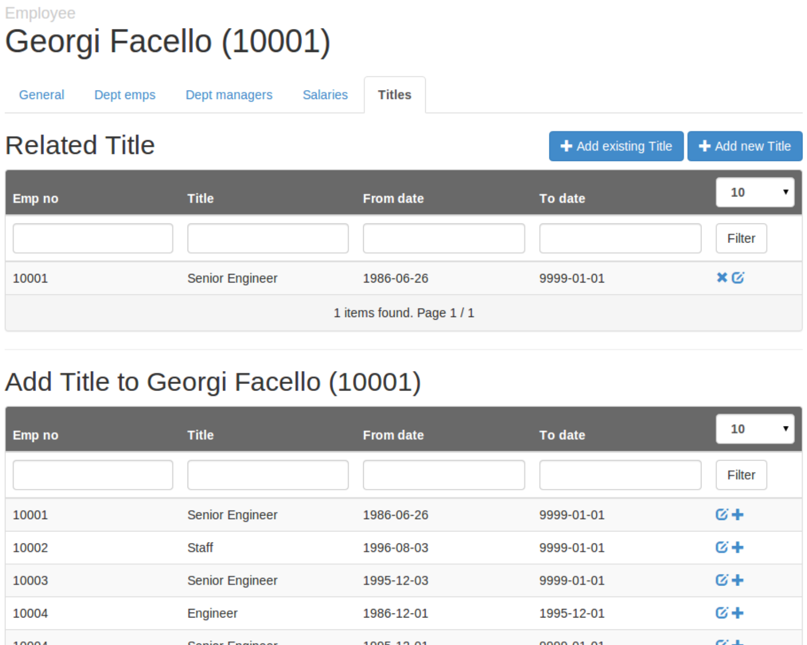
Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

Time 00:00

Hour

Minute

Relationship Title



Employee

Georgi Facello (10001)

General | Dept emps | Dept managers | Salaries | **Titles**

Related Title

+ Add existing Title   + Add new Title

Emp no	Title	From date	To date	10 ▾
				Filter
10001	Senior Engineer	1986-06-26	9999-01-01	✖ ✎

1 items found. Page 1 / 1

Add Title to Georgi Facello (10001)

Emp no	Title	From date	To date	10 ▾
				Filter
10001	Senior Engineer	1986-06-26	9999-01-01	✎ +
10002	Staff	1996-08-03	9999-01-01	✎ +
10003	Senior Engineer	1995-12-03	9999-01-01	✎ +
10004	Engineer	1986-12-01	1995-12-01	✎ +
10004	Senior Engineer	1995-12-01	9999-01-01	✎ +

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({..});
```

Im Normalfall verwendet man nur ein Schema in seiner Dancer-Anwendung:

Configuration

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

Es sind aber auch mehrere möglich:

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
```

Das Schema `legacy` wird dann wie folgt verwendet:

Multiple Schemas

```
use Dancer::Plugin::DBIC;

schema('legacy')->resultset('UserDb')->search({..});
```

Im Gegensatz zu Dancer::Plugin::Database bietet das DBIC-Plugin keine automatische Unterstützung für UTF-8. Also ist die entsprechende DBI-Option in der Konfiguration einzutragen, hier für MySQL:

UTF-8 for MySQL

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

Die Optionen für die gängigen Datenbanken in der Übersicht:

SQLite	`sqlite_unicode: 1`
MySQL	`mysql_enable_utf8: 1`
PostgreSQL	`pg_enable_utf8: 1`

Das DBIC-Plugin erzeugt dynamisch ein DBIx::Class::Schema, wenn die Schema-Klasse (`schema_class`) nicht angegeben wird. Dazu ist das Modul DBIx::Class::Schema::Loader erforderlich.
Dies ist nicht empfehlenswert für den Produktionseinsatz, jedoch praktisch für den TableEditor.

Create schema dynamically

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

Engines

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

Die Sessionengines werden in Dancer für gewöhnlich transparent für den Anwendungscode in der Konfiguration eingerichtet:

Configuration

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

Das ermöglicht es, auf dem Liveserver eine effizientere Engine zu verwenden (z.B. Storable) und auf dem Entwicklungsserver eine Engine, die einem beim debuggen hilft (z.B. YAML).
Die Optionen für Dancer::Session::DBIC ähneln der Konfiguration von Dancer::Plugin::DBIC, zusätzlich können wir festlegen wie die Sessions aus der Datenbank abgerufen werden können:

resultset	DBIx::Class resultset
id_column	primary key
data_column	field for session data

Das sieht dann z.B. für Interchange6::Schema (Version 0.015) so aus:

Configuration

```
session: "DBIC"
session_options:
  dsn: dbi:mysql:interchange6
  user: racke
  pass: nevairbe
  schema_class: Interchange6::Schema
  resultset: Session
  id_column: sessions_id
  data_column: session_data
session_expires: 12 hours
```

Die Konfiguration kann aber ebenso im Hauptmodul stattfinden:

Configuration

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

Folgendermaßen sieht die Tabelle `sessions` aus, die vom Schema Interchange6::Schema (Version 0.015) erzeugt wird:

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(@_); },
};
```

Beim Überschreiten der erlaubten Ablaufzeit wird die Sitzung ungültig, sie wird jedoch nicht in der Datenbank gelöscht. Dafür ist ein Skript zur regelmäßigen Löschung der abgelaufenen Datensätze erforderlich.
JSON andere DBIC connection? tests?

Session expiration

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

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

Features

- HTML editor
- Date and time picker
- File uploads

We are using Summernote, the “Super Simple WYSIWYG Editor on Bootstrap”, found at
`http://hackerwins.github.io/summernote/`.

WYSIWYG editor

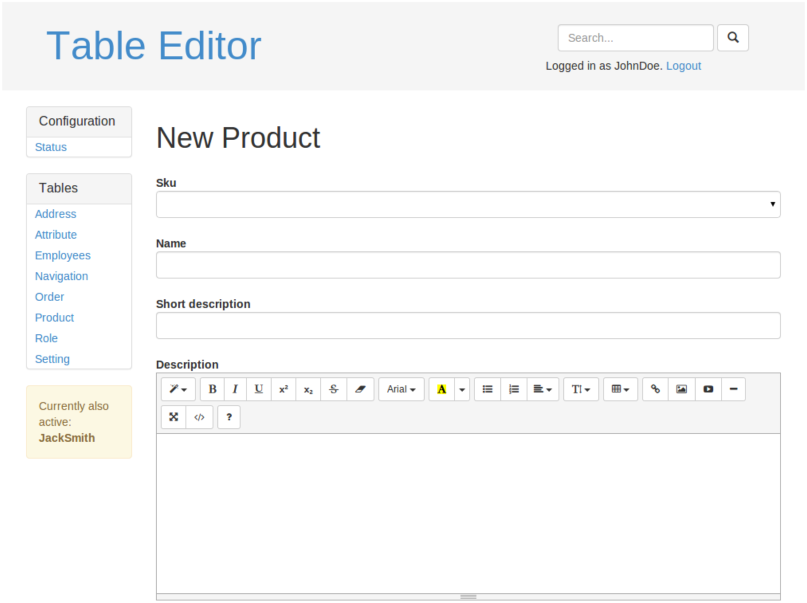


Table Editor

Search...

Logged in as JohnDoe. Logout

Configuration

Status

Tables

Address

Attribute

Employees

Navigation

Order

Product

Role

Setting

Currently also active: **JackSmith**

New Product

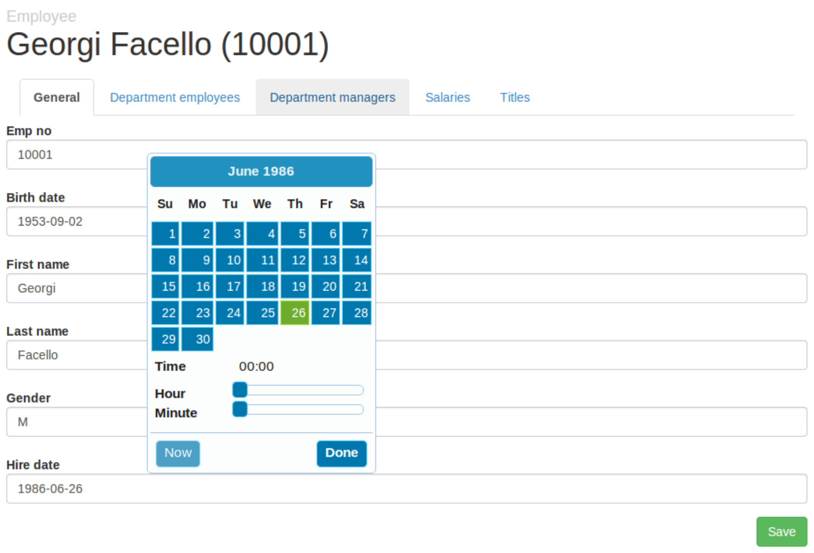
Sku

Name

Short description

Description

Date picker



Employee

Georgi Facello (10001)

General | Department employees | Department managers | Salaries | Titles

Emp no	10001
Birth date	1953-09-02
First name	Georgi
Last name	Facello
Gender	M
Hire date	1986-06-26

June 1986

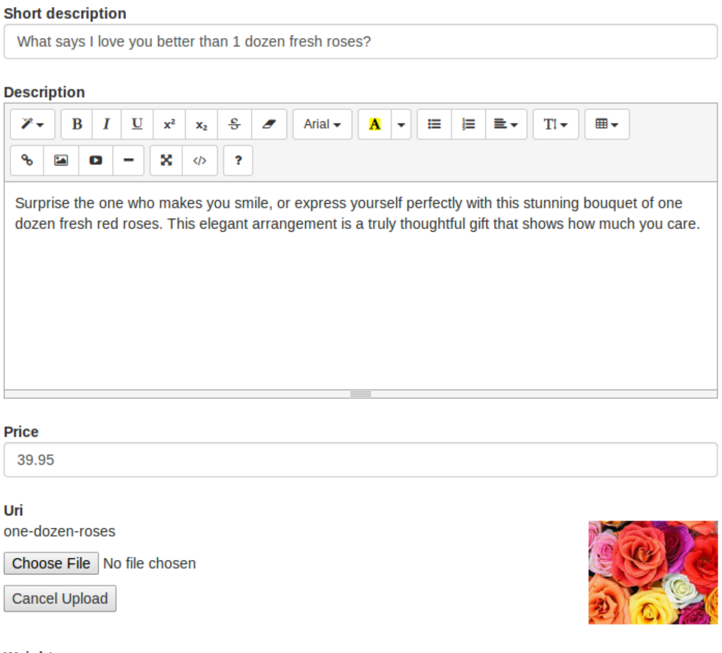
Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

Time 00:00

Hour

Minute

Image upload



Short description

What says I love you better than 1 dozen fresh roses?

Description

Surprise the one who makes you smile, or express yourself perfectly with this stunning bouquet of one dozen fresh red roses. This elegant arrangement is a truly thoughtful gift that shows how much you care.

Price

39.95

Uri

one-dozen-roses

Im günstigsten Fall kann die Installation mit 4 Schritten erledigt werden:

Installation

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

Driver

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

Das Frontend für den TableEditor ist mit Angular und Bootstrap erstellt. Das Theme kann sehr einfach durch Austausch der CSS-Datei für

Bootstrap geändert werden.

Routes

```
get '/:class/:id' => require_login sub {
    # retrieve database record and add relationships
    ...

    return to_json($data, {allow_unknown => 1});
};
```

Für die Integration von Authentifizierung in eine Dancer-Anwendung empfehlen wir wärmestens das Auth::Extensible Plugin.

Login

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

Beziehungen werden automatisch angezeigt.

Relationships

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

Filter

Es fehlen Felder in related orderline (Übersicht)

Different DBIC keys

Configuration

- Auth::Extensible
- DBIC
  - `default`

Configuration

```
TableEditor:
    classes:
        Media:
            columns:
                uri:
                  column_type: 'image_upload'
                  upload_dir: 'images/upload/media'
                  upload_max_size: 1000000
                  upload_extensions: [jpg, jpeg, gif, PNG]
```

Das Git-Repository für den TableEditor befindet sich auf Github:

Development

`https://github.com/interchange/TableEditor`

Was ist mit Dancer2 ?

Für Dancer2 existiert bereits ein Plugin:

`https://metacpan.org/pod/Dancer2::Plugin::DBIC`

Die Sessionengine und der TableEditor wurden noch nicht auf Dancer2 portiert.

Dancer2

Plugin::DBIC	`https://metacpan.org/pod/Dancer2::Plugin::DBIC`
Session::DBIC	`https://metacpan.org/pod/Dancer2::Session::DBIC`
TableEditor	not yet ported

`https://github.com/castaway/dbix-class-book`

Slides

Slides: `http://www.linuxia.de/talks/yapc2014/dancer-dbic-en-beamer.pdf`

Perl::Dancer Conference

`http://act.perl.dance/`

Perl::Dancer Conference