Perl Mongers im Ruhrgebiet

Ruhr.pm

Das Catalyst MVC-Framework

Autor: Simon Wilper

EMail: simon AT ruhr.pm.org

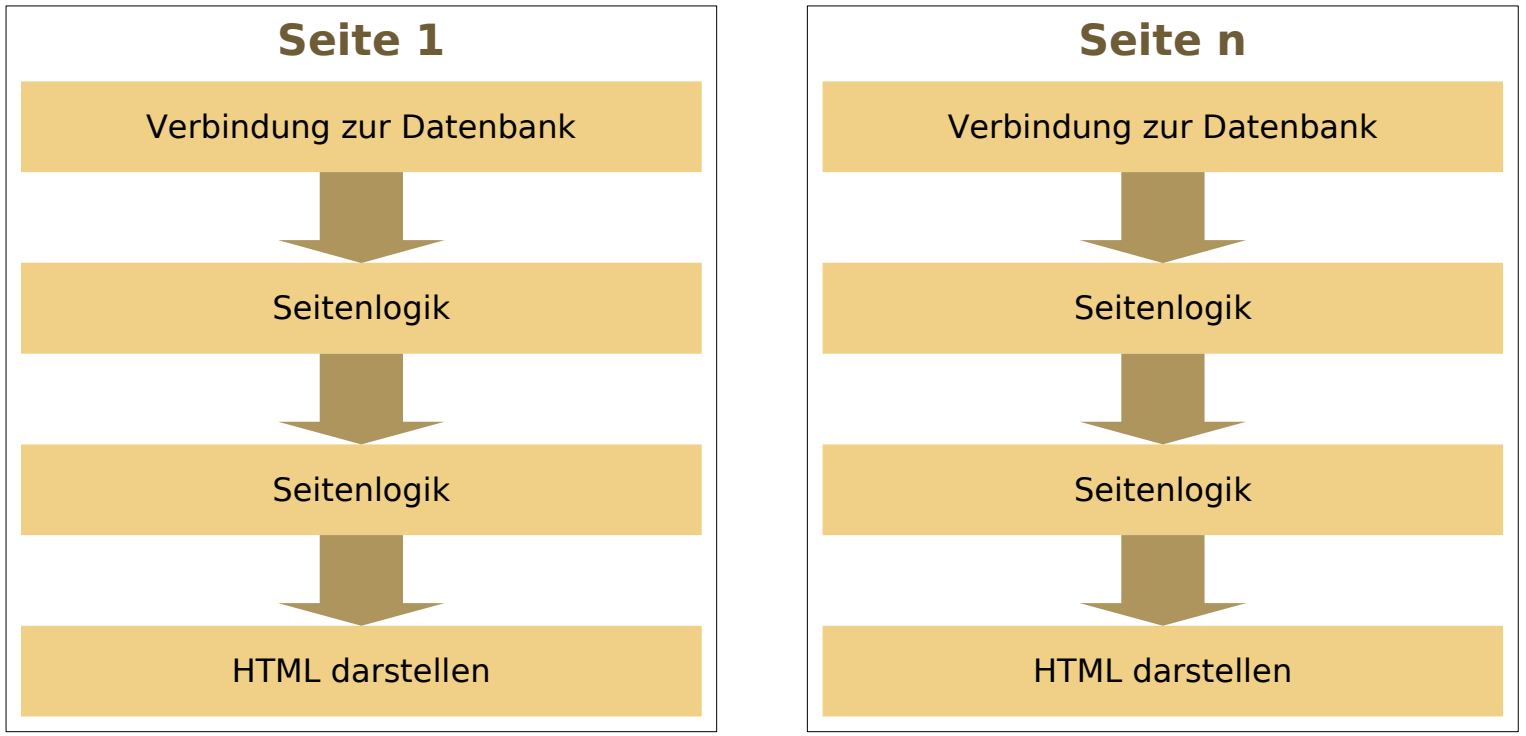
Datum: 12. August 2008

http://ruhr.pm.org/

Wozu MVC?

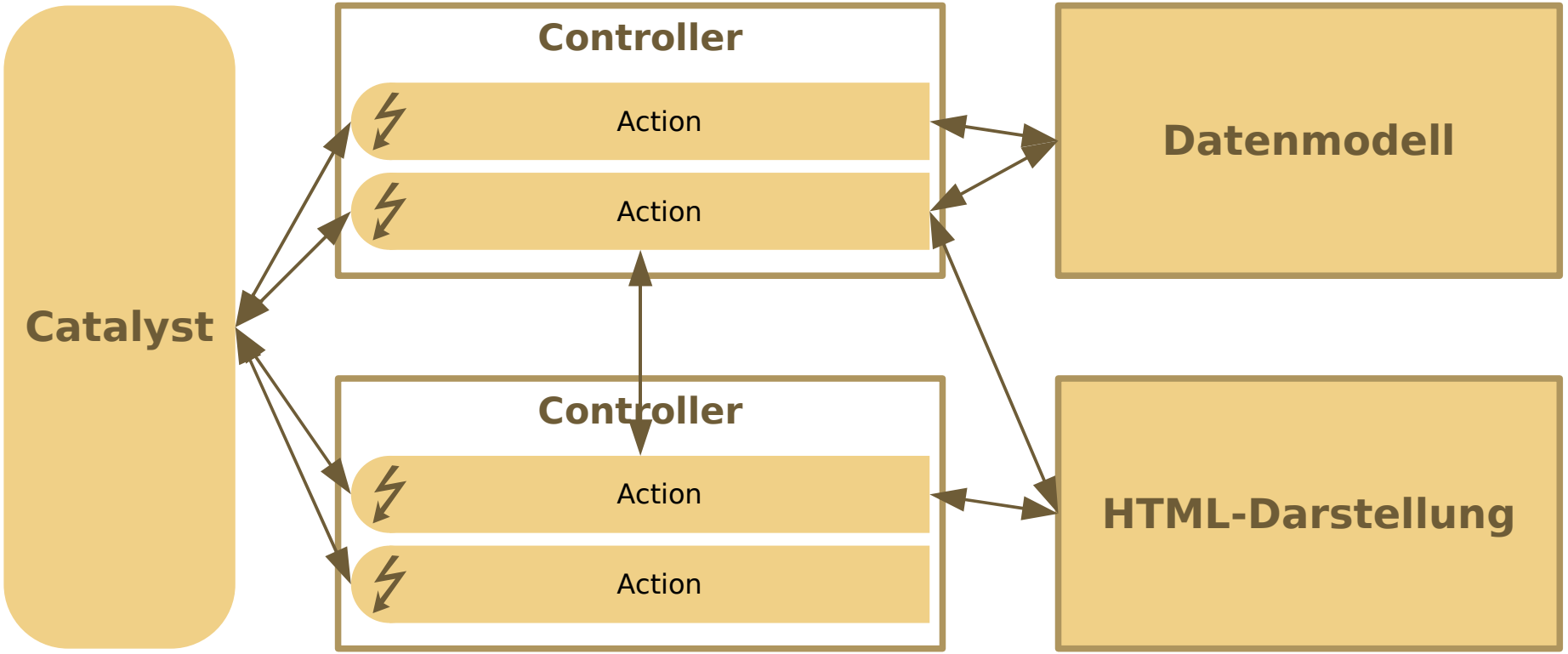
- Trennung in:
  - Datenmodell (Model)
  - Praesentation (View)
  - Steuerung (Controller)
- Hohe Wiederverwendbarkeit der Komponenten
- Einfache Erweiterbarkeit → Flexibilitaet

Umsetzung in Catalyst



Ruhr.pm

Umsetzung in Catalyst ctd.



Installation

- Voraussetzung: Perl 5.8.1 oder besser
- Catylist-Installation entweder via Package-Manager der Distribution
- oder CPAN:
  - cpan Catalyst::Runtime Catalyst::Devel
- Zusaetzlich:
  - TT (Template Toolkit) als Template-System: Catalyst::View::TT
  - FormBuilder

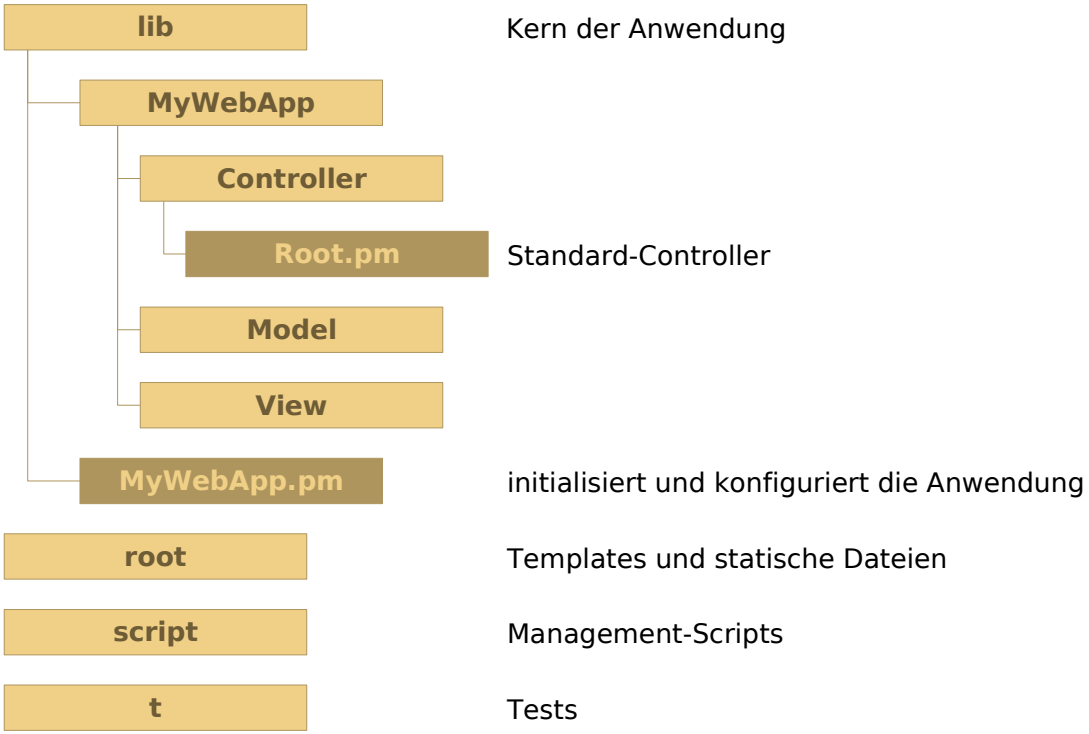
Erstellen einer Webanwendung

- Erstellen des Skeletons
- Praesentationstemplates erstellen
- Controller mit Actions hinzufuegen
- Datenmodell hinzufuegen

Skeleton erstellen

```
$ catalyst.pl MyWebApp
```

catalyst.pl legt eine Standard- Verzeichnisstruktur mit allen benoetigten Dateien und Standalone-Development-Umgebung an.



Kern der Anwendung

Standard-Controller

initialisiert und konfiguriert die Anwendung

Templates und statische Dateien

Management-Scripts

Tests

Ruhr.pm

Erster Test: script/mywebapp_server.pl

- Standalone Server fuer die Entwicklung
- Optionen:

```
-d      debug:          im Debug-Mode starten
-f      fork:           Jede Anfrage in einem neuen Prozess starten
-p      port:           Port des Standalone-Servers (Standard: 3000)
-r      restart:        Automatisch neu starten, wenn sich etwas im
                        Anwendungskern geaendert hat.
-rd     restartdelay:   Zeitspanne zwischen den Ueberpruefungen
-rr     restartregex:   Regulaerer Ausdruck, der auf geaenderte Dateien zutrifft,
                        wann neugestartet werden soll
```

View und Templates hinzufuegen

View-Modul erstellen

```
> script/mywebapp_create.pl view TT TT
```

- `script/mywebapp_create.pl` — Create-Script zur Erstellung von neuen MVC-Modulen
- `view` — Typ, der erstellt werden soll (model/view/controller)
- `TT` — Name des Moduls innerhalb der Webanwendung
- `TT` — Name des Catalyst Helpers

TT-Template erstellen: MyWebApp/root/index.tt

```
<div id="contents">
        [% message | html %]
</div>
```

- `[% message | html %]` — Uebergebene Variable „message“ einfuegen, „html“-Modifier: HTML-Tags escapen

Controller hinzufuegen

Controller-Modul erstellen

```
> script/mywebapp_create.pl controller Welcome
```

Name des Controller-Moduls

Actions im Controller anpassen: lib/MyWebApp/Controller/Welcome.pm

```
sub index :Global {
    my ( $self, $c, @args ) = @_;

    $c->response->body('Matched
        MyWebApp::Controller::Welcome in
        Welcome: '. $args[0]);
}
```

$**self**: Controller selber (siehe OO-Perl)

$**c**: Catalyst-Context

@**args**: Pfad-Argumente

$c->response->body: Direkt Text in den Responsebody schreiben

Template benutzen

Template dem „Stash“ hinzufuegen:

```
$c->stash->{title} = 'Willkommen';
$c->stash->{message} = 'Hallo Welt!';

$c->stash->{template} = 'welcome.tt';
```

„**title**“, „**message**“: Variablen in der Template „welcome.tt“

„**template**“: Variable, die die verwendete Vorlage haelt

Das Request-Objekt

Beispiel: Alle Request-URL-Parameter auflisten:

Im Controller:

```
$c->stash->{params} = $c->request->parameters;
```

Hash-Referenz auf alle Request-Parameter, aus dem Request-Objekt speichern.

Im Template:

```
[% IF params.size() > 0 %]
<ul>
        [% FOREACH element IN params %]
        <li>[% element.key %] =
                [% element.value %]</li>
        [% END %]
</ul>
[% ELSE %]
<p>No params</p>
[% END %]
```

 nur anzeigen, wenn sich Elemente im Hash befinden

FOREACH-Schleife iteriert ueber alle Elemente, mit key()- und value()-Member auf die Paare zugreifen

Model hinzufuegen

Model auf „Plain DBI“-Basis erstellen

```
> script/mywebapp_create.pl model MyDataModel DBI
```

MyDataModel: Name des Models

DBI: DBI-Model erstellen

Alle Benutzernamen auflisten

Im Model:

```
sub simple_list {
        my( $self ) = @_;
        my $dbh = $self->dbh;
        return $dbh->selectall_hashref(
                'SELECT * FROM users', 'id'
        );
}
```

DB-Handle wird mit $self uebergeben

Hash-Referenz von allen Benutzern mit Key = Tabellenfeld 'id'

Model hinzufuegen ctd.

Im Controller:

```
$c->stash->{users} =
        $c->model('MyDataModel')->simple_list();
```

Member „simple_list()“ aufrufen und in Stash-Variablen 'users' speichern.

Im Template:

```
<ul>
[% FOREACH user IN users %]
        <li>[% user.key %]:
        [% user.value.lastname %],
        [% user.value.firstname %]</li>
[% END %]
</ul>
```

Auf DB-Felder ueber '.' zugreifen

Da eine ganze Hash-Referenz uebergeben wurde, muss immer ueber 'key'/'value' auf die Felder zugegriffen werden.

Formulare mit FormBuilder

- automatische Validierung
  - JavaScript
  - intern
- Formulare mit YAML erstellen

```
name: add_user
method: post
fields:
        firstname:
                label: First Name
                type: text
                size: 30
                required: 1

        lastname:
                label: Last Name
                type: text
                size: 30
                required: 1
```

Formulare mit FormBuilder ctd.

Im Controller:

```
sub add :Global Form {
        my ( $self, $c ) = @_;

        my $form = $self->formbuilder;

        if ( $form->submitted && $form->validate ){
                my $rc = $c->model('MyDataModel')
                        ->addNewUser( $form );
                $c->response->body( 'RC: '. $rc );
        }
}
```

In der Vorlage:

```
[% FormBuilder.render %]
```

:Form-Attribut sucht nach:
root/forms/$MODULNAME/add.fb

Und Formular
root/$MODULNAME/add.tt

Ruhr.pm

FormBuilder dynamisch beeinflussen

```
<head>
<title>[% formbuilder.title %]</title>
  [% formbuilder.jshead %]<!-- javascript -->
</head>
 <body>
  [% formbuilder.start -%]
  <div id="form">
    [% FOREACH field IN formbuilder.fields -%]
    <p>
        <label>
  <span [% IF field.required %]class="required"[%END%]>[%field.label%]</span>
        </label>
      [% field.field %]
      [% IF field.invalid -%]
          <span class="error">
              Missing or invalid entry, please try again.
          </span>
      [% END %]
      </p>
    [% END %]
    <div id="submit">[% formbuilder.submit %]</div>
    <div id="reset">[% formbuilder.reset %]</div>
    </div>
  </div>
  [% formbuilder.end -%]
</body>
```

Deployment am Beispiel Apache

- httpd.conf anpassen

```
PerlSwitches -I/.../MyWebApp/lib
PerlModule MyWebApp

<Location />
        SetHandler modperl
        PerlResponseHandler MyWebApp
</Location>
```