

Code macht Daten

Migrations und Fixtures im Contao-Kontext

Problemstellung allgemein

Hakelige Lage:

Änderung im Code funktioniert nur mit
Änderung in Datenbank zusammen

Wer?

Alex Wuttke, 34, Entwickler, Rostock

Insgesamt 10+ Jahre Contao-Entwicklung,
angefangen 2012

Selbständig tätig, meist für Agenturen,
in Contao, Frontend, Webanwendungen,
Tech-Beratung



Typische Beispiele

Template umbenennen


form_textarea_foo → form_textarea_bar

Template umbenennen

form_textarea_foo → form_textarea_bar

Symfony Exception Symfony Docs

Exception HTTP 500 Internal Server Error

Could not find template "form_textarea_foo" 

Exception Logs **1** Stack Trace

Exception [-]

- + in vendor/contao/core-bundle/src/Resources/contao/library/Contao/TemplateLoader.php (line 156)
- + in vendor/contao/core-bundle/src/Resources/contao/library/Contao/TemplateLoader.php :: **getDefaultPath** (line 112)
- + in vendor/contao/core-bundle/src/Resources/contao/library/Contao/Controller.php :: **getPath** (line 95)
- + in vendor/contao/core-bundle/src/Resources/contao/library/Contao/TemplateInheritance.ph

Template umbenennen

Technischer Schritt „Umbenennen des Templates“ ist
Umbenennen der Datei + Änderung in der Datenbank

Zwischenbemerkung zum Ziel

Wir möchten:

Deployment einer Sache passiert in einem Zug

Code für ein Feature oder eine Änderung ist eine Einheit

Frontend-Module konfigurieren

Neues Modul – selber gebaut oder neue Konfiguration

Frontend-Module konfigurieren

Titel und Typ

Titel*
Nachrichtenliste - Stil: Slider
Bitte geben Sie den Titel des Moduls ein.

Überschrift
 h2 ▼
Hier können Sie dem Modul eine Überschrift hinzufügen.

Modultyp
Nachrichtenliste
Bitte wählen Sie den Typ des Moduls.

Modul-Konfiguration

Nachrichtenarchive*
 Alle auswählen
 Aktuelle Termine
 Aktuelles hervorgehoben
 Instagram
 Pressemeldungen
 Stellenausschreibungen
Bitte wählen Sie ein oder mehrere Nachrichtenarchive.

Nachrichtener Leser
 - ▼
Automatisch zum Nachrichtener Leser wechseln, wenn ein Beitrag ausgewählt wurde.

Hervorgehobene Beiträge
 Alle Beiträge anzeigen ▼
Hier legen Sie fest, wie hervorgehobene Beiträge gehandhabt werden.

Elemente überspringen
 0 ▼
Hier legen Sie fest, wie viele Elemente übersprungen werden sollen.

Anzahl an Elementen*
 0 ▼
Hier können Sie die Gesamtzahl der Elemente begrenzen. Geben Sie 0 ein, um alle anzuzeigen.

Sortierreihenfolge
 Datum absteigend ▼
Hier können Sie die Sortierreihenfolge festlegen.

Elemente pro Seite
 0 ▼
Die Anzahl an Elementen pro Seite. Geben Sie 0 ein, um den automatischen Seitenumbruch zu

Template-Einstellungen

Meta-Felder
 Alle auswählen
 Datum
 Autor
 Kommentare
Hier können Sie die Meta-Felder auswählen.

Nachrichten-Template
 news_latest_slider (XYZ) ▼
Hier können Sie ein Nachrichten-Template auswählen.

Modul-Template
 mod_newslist_slider (XYZ) ▼
Hier können Sie das Modul-Template auswählen.

Bildeinstellungen

Bildgröße
 - ▼
Hier können Sie die Abmessungen des Bildes und den Skalierungsmodus festlegen.

Zugriffsschutz

Modul schützen
Das Modul nur bestimmten Mitgliedergruppen anzeigen.

Experteneinstellungen

Nur Gästen anzeigen
Das Modul verschwindet, sobald ein Mitglied angemeldet ist.

ID/CSS-Klasse

Hier können Sie eine ID und beliebig viele Klassen eingeben.

Frontend-Module konfigurieren

Okay, vielleicht nicht ganz so manuell

Weitere Fälle

Extensions, die ihre Config in der Datenbank speichern

Contao-Basis-Setup mit Daten

Formulargenerator + Formular-Hook

Fazit

Neuer Code → Neue Daten

Ergibt sich aus der Entwicklung

Lösungen

Bestehende Daten(-Eigenschaften/-Strukturen)
aufgrund einer bestimmten Regel ändern

→ Migrations

Konkrete Daten neu erzeugen / festlegen,
die für etwas benötigt werden

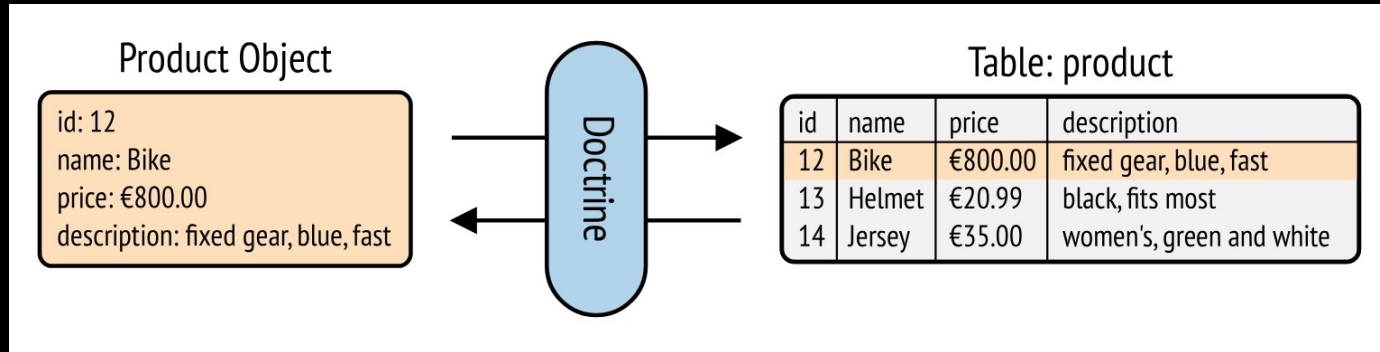
→ Fixtures

Ein Blick auf Symfony+Doctrine

Definition der Datenstruktur in Doctrine Entities

Struktur kommt über Migrations in die Datenbank

Doctrine Entities



Doctrine Entities

```
// src/Entity/Product.php
namespace App\Entity;

use App\Repository\ProductRepository;
use Doctrine\ORM\Mapping as ORM;

#[ORM\Entity(repositoryClass: ProductRepository::class)]
class Product
{
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column]
    private ?int $id = null;

    #[ORM\Column(length: 255)]
    private ?string $name = null;

    #[ORM\Column]
    private ?int $price = null;

    public function getId(): ?int
    {
        return $this->id;
    }

    // ... getter and setter methods
}
```

Migrations

doctrine:migrations:diff →

```
Version20210820150551.php
Version20210909223855.php
Version20210927094846.php
Version20211007105840.php
Version20211209232626.php
```






```
final class Version20210820150551 extends AbstractMigration
{
    public function getDescription(): string
    {
        return '';
    }

    public function up(Schema $schema): void
    {
        // this up() migration is auto-generated, please modify it to your needs
        $this->addSql('CREATE TABLE card (id INT AUTO_INCREMENT NOT NULL, product VARCHAR(255) NOT NULL, elements LONGTEXT DEFAULT NULL COMMENT \'(DC2Type:array)\' PRIMARY KEY(id)) DEFAULT CHARACTER SET utf8mb4 COLLATE \'utf8mb4_unicode_ci\' ENGINE=InnoDB');
        $this->addSql('CREATE TABLE invite (id INT AUTO_INCREMENT NOT NULL, card_id INT NOT NULL, PRIMARY KEY(id)) DEFAULT CHARACTER SET utf8mb4 COLLATE \'utf8mb4_unicode_ci\' ENGINE=InnoDB');
        $this->addSql('CREATE TABLE invite_user (invite_id INT NOT NULL, user_id INT NOT NULL, PRIMARY KEY(invite_id, user_id)) DEFAULT CHARACTER SET utf8mb4 COLLATE \'utf8mb4_unicode_ci\' ENGINE=InnoDB');
        $this->addSql('CREATE TABLE product (id INT AUTO_INCREMENT NOT NULL, name VARCHAR(255) NOT NULL, DEFAULT CHARACTER SET utf8mb4 COLLATE \'utf8mb4_unicode_ci\' ENGINE=InnoDB');
        $this->addSql('CREATE TABLE user (id INT AUTO_INCREMENT NOT NULL, username VARCHAR(180) NOT NULL, email VARCHAR(180) NOT NULL, password VARCHAR(255) NOT NULL, is_verified TINYINT(1) NOT NULL, PRIMARY KEY(id)) DEFAULT CHARACTER SET utf8mb4 COLLATE \'utf8mb4_unicode_ci\' ENGINE=InnoDB');
        $this->addSql('ALTER TABLE card ADD CONSTRAINT FK_161498D34584665A FOREIGN KEY (product_id) REFERENCES product (id) ON DELETE CASCADE');
        $this->addSql('ALTER TABLE card ADD CONSTRAINT FK_161498D37E3C61F9 FOREIGN KEY (card_id) REFERENCES card (id) ON DELETE CASCADE');
        $this->addSql('ALTER TABLE invite ADD CONSTRAINT FK_C7E210D74ACC9A20 FOREIGN KEY (card_id) REFERENCES card (id) ON DELETE CASCADE');
        $this->addSql('ALTER TABLE invite_user ADD CONSTRAINT FK_95A717C3EA4177 FOREIGN KEY (invite_id) REFERENCES invite (id) ON DELETE CASCADE');
        $this->addSql('ALTER TABLE invite_user ADD CONSTRAINT FK_95A717C3A76ED3 FOREIGN KEY (user_id) REFERENCES user (id) ON DELETE CASCADE');
    }

    public function down(Schema $schema): void
    {
        // this down() migration is auto-generated, please modify it to your needs
        $this->addSql('ALTER TABLE invite DROP FOREIGN KEY FK_C7E210D74ACC9A20');
        $this->addSql('ALTER TABLE invite_user DROP FOREIGN KEY FK_95A717C3EA4177');
        $this->addSql('ALTER TABLE card DROP FOREIGN KEY FK_161498D34584665A');
        $this->addSql('ALTER TABLE card DROP FOREIGN KEY FK_161498D37E3C61F9');
        $this->addSql('ALTER TABLE invite_user DROP FOREIGN KEY FK_95A717C3A76ED3');
        $this->addSql('DROP TABLE card');
        $this->addSql('DROP TABLE invite');
        $this->addSql('DROP TABLE invite_user');
        $this->addSql('DROP TABLE product');
        $this->addSql('DROP TABLE user');
    }
}
```

Migrations

doctrine:migrations:migrate

-  Version20210820150551.php
-  Version20210909223855.php
-  Version20210927094846.php
-  Version20211007105840.php
-  Version20211209232626.php

Migrations in Contao

DCAs

contao:migrate

```
* DROP TABLE tl_columnset
* DROP TABLE tl_cron
* DROP TABLE tl_extension
* DROP TABLE tl_metamodel
* DROP TABLE tl_metamodel_attribute
* DROP TABLE tl_metamodel_dca
* DROP TABLE tl_metamodel_dca_combine
* DROP TABLE tl_metamodel_dcasetting
* DROP TABLE tl_metamodel_filter
* DROP TABLE tl_metamodel_filtersetting
* DROP TABLE tl_metamodel_rating
* DROP TABLE tl_metamodel_rendersetting
* DROP TABLE tl_metamodel_rendersettings
* DROP TABLE tl_metamodel_tabletext
* DROP TABLE tl_metamodel_tag_relation
* DROP TABLE tl_metamodel_translatedcheckbox
* DROP TABLE tl_metamodel_translatedlongblob
* DROP TABLE tl_metamodel_translatedlongtext
* DROP TABLE tl_metamodel_translatedtabletext
* DROP TABLE tl_metamodel_translatedtext
* DROP TABLE tl_newsletter_recipients_backup
* DROP TABLE tl_repository_installs
* DROP TABLE tl_repository_instfiles
* DROP TABLE tl_session
```

Okay. Wozu?

Migrations nicht nur automatisch,
sondern beliebige Änderungsmöglichkeiten

Bestehende Daten

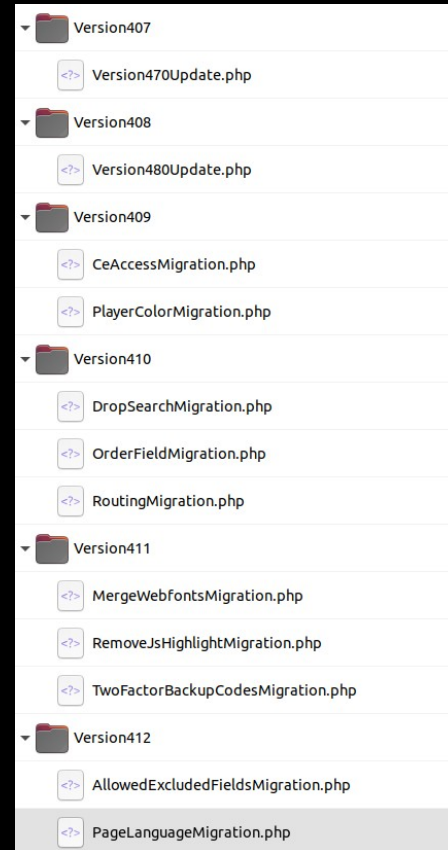
Datenstrukturen

Dateneigenschaften

Contao Migration Framework

Contao Migration Framework

core-bundle: src/Migration



Contao Migration Framework

```
class PageLanguageMigration extends AbstractMigration
{
    // ...

    public function shouldRun(): bool
    {
        $schemaManager = $this->connection->createSchemaManager();

        if (!$schemaManager->tablesExist(['tl_page'])) {
            return false;
        }

        $pageColumns = $schemaManager->listTableColumns('tl_page');

        if (!isset($pageColumns['language'])) {
            return false;
        }

        $count = $this->connection->fetchOne("
            SELECT
                COUNT(*)
            FROM
                tl_page
            WHERE
                type='root' AND SUBSTRING(language, 3, 1) = '-'
        ");

        return $count > 0;
    }

    public function run(): MigrationResult
    {
        $pages = $this->connection->fetchAllAssociative("
            SELECT
                id, language
            FROM
                tl_page
            WHERE
                type='root' AND SUBSTRING(language, 3, 1) = '-'
        ");

        foreach ($pages as $page) {
            $this->connection->update(
                'tl_page',
                ['language' => LocaleUtil::canonicalize($page['language']),
                ['id' => $page['id']]
            );
        }

        return $this->createResult(true);
    }
}
```


Contao Migration Framework

```
namespace App\Migration;

use Contao\CoreBundle\Migration\AbstractMigration;
use Contao\CoreBundle\Migration\MigrationResult;
use Doctrine\DBAL\Connection;

class FooBarMigration extends AbstractMigration
{
    public function __construct(private readonly Connection $connection)
    {
    }

    public function shouldRun(): bool
    {
    }

    public function run(): MigrationResult
    {
    }
}
```

Migrations in Bundles

Bisheriges Beispiel App-spezifisch

Funktioniert in Contao Bundles genauso

Bitte baut Migrations

Automatisch

contao:migrate in den build+deploy-Vorgang

Fixtures

(neue) Daten in die Datenbank bringen

`doctrine/doctrine-fixtures-bundle`

Fixtures: Symphony-Denke

Use Case Dummy-Daten

Symphony denkt weniger in Contao-artigen Fällen

Fixtures: Code

```
namespace App\DataFixtures;

use App\Entity\Product;
use Doctrine\Bundle\FixturesBundle\Fixture;
use Doctrine\Persistence\ObjectManager;

class AppFixtures extends Fixture
{
    public function load(ObjectManager $manager)
    {
        $product = new Product();
        $product->setName('Foo');
        $product->setPrice(123);

        $manager->persist($product);

        $manager->flush();
    }
}
```

Fixtures: Code

```
namespace App\DataFixtures;

use Contao\ModuleModel;
use Doctrine\Bundle\FixturesBundle\Fixture;

class AppFixtures extends Fixture
{
    public function load()
    {
        $module = new ModuleModel();
        $module->setRow([
            /*...*/
        ]);
        $module->save();
    }
}
```

Fixtures laden

`doctrine:fixtures:load`

AbEr

Default: Alle Daten löschen, Fixtures rein laden

Vgl. Dummy-Daten-Gedanke

Manchmal hilfreich

Wenn nicht: --append

Fixtures: Append

Wichtig: Prüfen, ob schon da

In Fixture selbst Tabelle purgen geht auch

→ Überschreiben und Anhängen mischen

Oder: `--purge-exclusions`

Hat das jetzt was gebracht?

Noch nicht ganz.

Manuelle Tätigkeit von „nach Deployment“ zu „vor dem Einchecken“ verschoben

Helferlein wären hilfreich

Helferlein Ansatz 1

Eingabe: Modelname+Daten

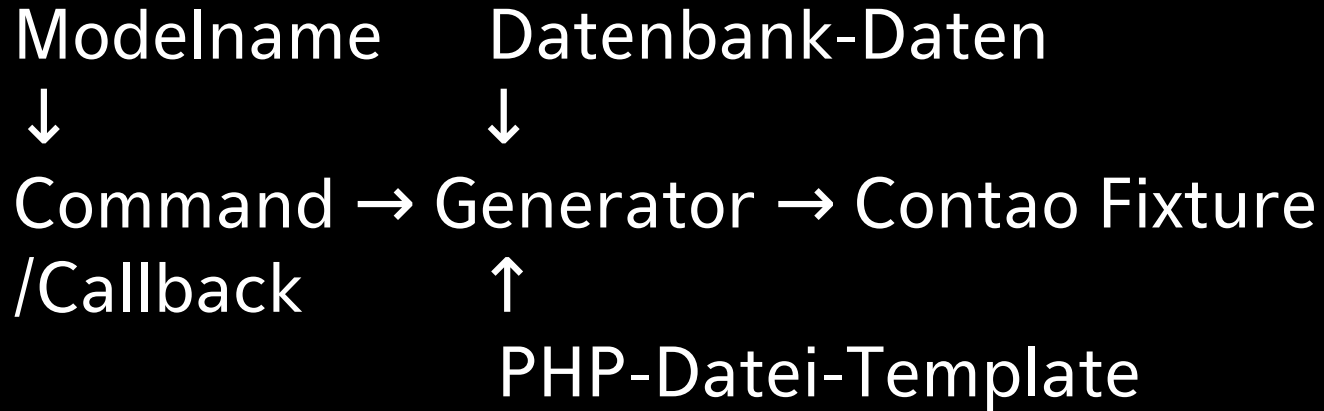


Command → Generator → Contao Fixture



PHP-Datei-Template

Helferlein Ansatz 2



Grenzfälle

Gemischte Nutzung von Super-Redaktion und Dev

Mögliche Lösung:

Ansätze aus der Sync- und Deploy-Praxis

Was stattdessen?

Alles, was über config geht, z. B. Bildgrößen

Nicht-persistent erzeugen (Content, Modules)

Theme-Bereich: contao-theme-framework

Formulare: Kommt drauf an

→ Haste, Symfony, DC_General Frontend, ConfigForms

