portrino

Websites und Kampagnen, die performen

AUTOMATISCHE CONTENT MIGRATIONS

Wie bekomme ich die Login Seite auf den LIVE?

portrino

AGENDA

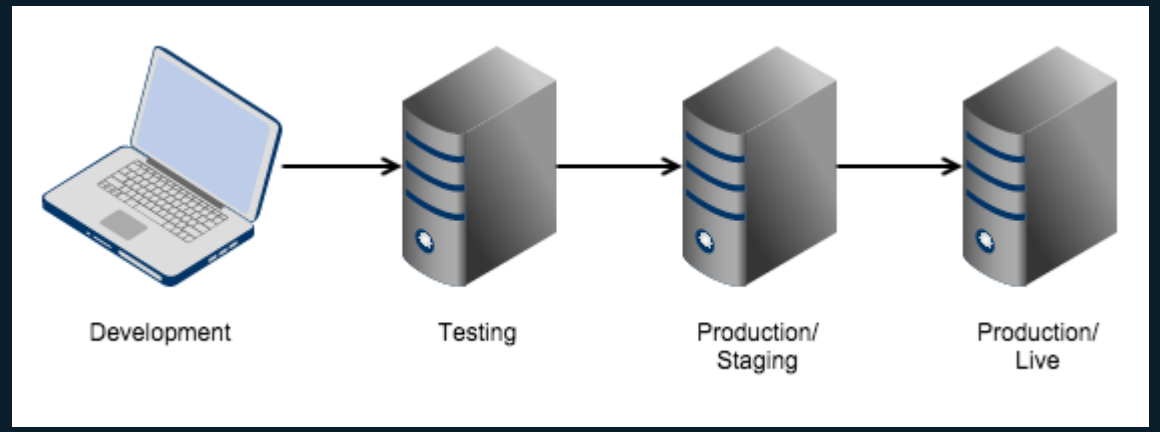
1. Ausgangslage
2. Anforderung
3. Lösung Schritt 1
4. Lösung Schritt 2
5. ~~Nachteile~~ Chancen

portrino

AUSGANGSLAGE

Mehrere Systeme

- LIVE (Redakteure pflegen)
- STAGING (Redakteure testen)
- DEV (PMs testen)
- Local-Dev-1 (Entwickler)
- Local-Dev-2
- Local-Dev-Drölf



ANFORDERUNG

- DEV baut ein Feature, das Seiten/Sysfolder/Content mit festen UIDs benötigt
- Lokal alles fertig anlegen
- automatisch auf allen Systemen ausrollen:
  - andere Lokale Entwicklungsumgebung
  - Test- und Staging-Systeme
  - Produktiv-System

LÖSUNG: SCHRITT 1

UIDs einzigartig auf allen beteiligten Systemen

EXT:px_dbsequencer

https://github.com/portrino/px_dbsequencer

https://extensions.typo3.org/extension/px_dbsequencer

- Extension seit 2014 bei GitHub
- schon vorher im SVN bei uns
- seit über 10 Jahren in jedem Projekt dabei
- aktuell bis v13 kompatibel

Lösung Schritt 1 | UIDs einzigartig (so wie du!)

portrino

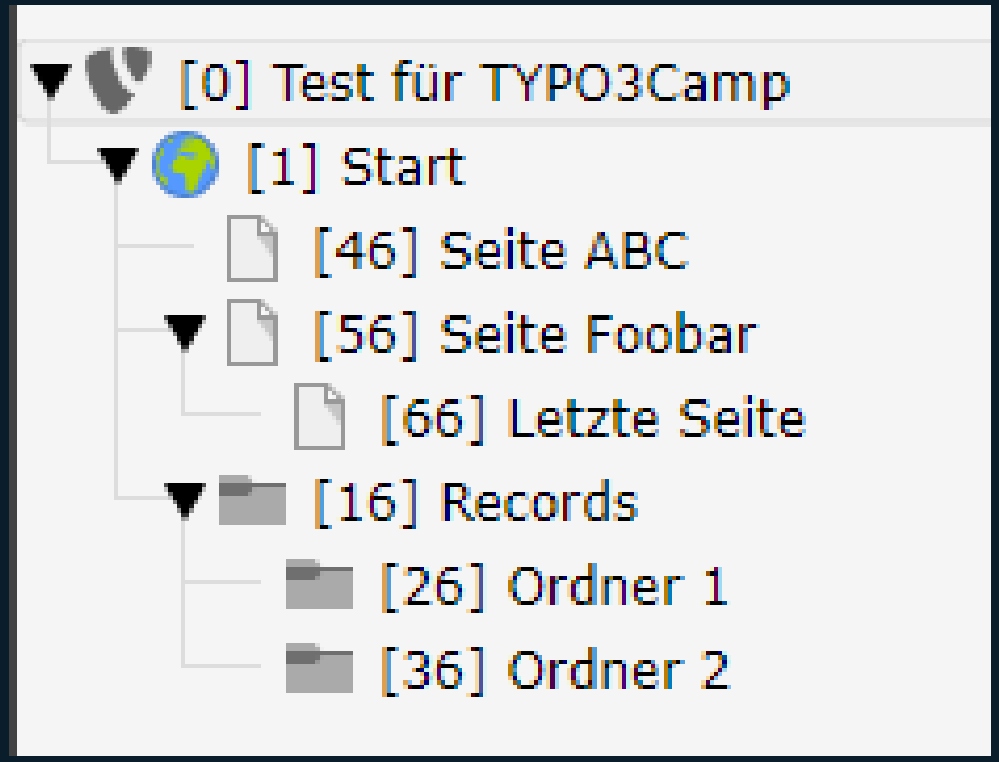
```
LocalConfiguration.php

'px_dbsequencer' => [
    'offset' => 10,
    'system' => '1',
    'tables' => 'pages,sys_category,tt_content,
                 tx_bootstrappackage_accordion_item,
                 tx_powermail_domain_model_field',
],
```

- **Sequenziert UIDs in festgelegten Schritten**
- **Tabellen festlegen:**
  z.B. `pages+tt_content`
  (müssen **`uid`** und **`TCA`** haben)
- **Offset festlegen:**
  wie viele Systeme maximal, z.B. „10“
- **aktuelles System festlegen:**
  zB „1“ für LIVE System

portrino

WAS MACHT DAS?



Lösung Schritt 1 | UIDs einzigartig (so wie du!)

portrino

AdditionalConfiguration.php

```php
switch ($applicationContext[0]) {
  case 'Production':
    $GLOBALS['TYPO3_CONF_VARS']['EXTENSIONS']['px_dbsequencer']['system'] = '1';
    break;
  case 'Development':
    if (array_key_exists(1, $applicationContext)) {
      switch ($applicationContext[1]) {
        case 'Local':
          if (array_key_exists(2, $applicationContext)) {
            switch ($applicationContext[2]) {
              case 'AB':
                $GLOBALS['TYPO3_CONF_VARS']['EXTENSIONS']['px_dbsequencer']['system'] = '4';
                $GLOBALS['TYPO3_CONF_VARS']['SYS']['sitename'] = $sitename . ' (AB)';
                break;
              case 'TG':
                $GLOBALS['TYPO3_CONF_VARS']['EXTENSIONS']['px_dbsequencer']['system'] = '6';
                $GLOBALS['TYPO3_CONF_VARS']['SYS']['sitename'] = $sitename . ' (TG)';
                break;
            }
          }
          break;
      }
    }
    break;
}
```

Lösung Schritt 1 | UIDs einzigartig (so wie du!)

portrino

```
# override.yaml

EXTENSIONS:
    px_dbsequencer:
        system: '%env(DB_SEQUENCER_SYSTEM_ID)%'
```

portrino

VORTEILE

- Inhalte können gefahrlos zwischen Systemen migriert werden
- UIDs können im Code (PHP, YAML, Typoscript) fixiert werden
- nachträglich erkennbar, in welchem System ein Datensatz angelegt wurde

LÖSUNG: SCHRITT 2

automatische Migrations bei jedem Deploy

portrino

EXT:migrator

https://github.com/portrino/px_dbmigrator

- Extension seit 2013 bei GitHub
- Kompatibel bis v13

Lösung Schritt 2 | automatische Migrations

portrino

LocalConfiguration.php

```
'migrator' => [
    'migrationFolderPath' => '../migrations',
    'mysqlBinaryPath' => '/usr/bin/mysql',
    'typo3cmsBinaryPath' => './bin/typo3cms',
],
```

- **Pfad zu den Migrations-Dateien**
  relativ zum Webroot
- **Pfad zur MySQL Binary**
- **Pfad zur TYPO3 Binary**
- **BE User „_cli_“**

Lösung Schritt 2 | automatische Migrations

portrino

```
{
  "name": "portrino/lexsoft-import",
  "type": "project",
  "require": {
    "typo3/cms-base-distribution": "^12.1",
  "scripts": {
    "post-autoload-dump": [
      "post-script:database-migrations"
    ],
    "post-script:database-migrations": [
      "./bin/typo3cms migration:migrateall",
      "./bin/typo3cms database:updateschema *.add,*.change"
  }
}
```

portino

FUNKTION

- nummerierte SQL Dateien (001.sql, 002.sql ...)
- werden in Reihenfolge importiert
- nach dem erfolgreichen Ausführen in **`sys_registry`** gespeichert
- wir benutzen TIMESTAMPS + kurze Definition, z.B.
    - 1556572658-pages-content-login.sql
    - 1678895310-sys-folder-news.sql
- Start über CommandLine oder Scheduler Task
- auch .sh möglich um z.B. Dateien zu verschieben und bash Befehle auszuführen

portrino

ABLAUF

- Lokal Seiten und Content anlegen
- SQL Dateien erzeugen (Adminer/phpmyadmin)
  (nur mit den tatsächlichen Änderungen, kein voller DUMP)
- in Ordern **`migrations`** ablegen als **`TIMESTAMP-inhalt.sql`**
- wird z.B. beim nächsten **`composer install`** ausgeführt

portrino

VORTEILE

- Migrations können manuell oder automatisch ausgeführt werden
- wichtige Seiten etc sind überall garantiert
- Deploy von neuen Features schneller und keine Fehler durch „vergessen“ von Inhalten
- durch `composer install` im normalen Workflow enthalten

NACHTEILE

Was geht nicht und wo muss man aufpassen?

portrino

NACHTEILE

- Kein undo
- wenn sich Tabellen ändern (Spalten entfernt) können alte Migrations fehlschlagen
- px_dbsequencer kann kein FAL
  also keine Bilder möglich

FRAGEN

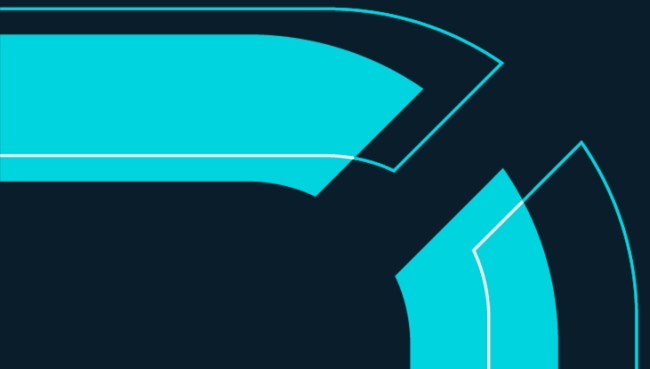
Wie macht ihr das?

portrino

Websites und Kampagnen, die performen

Thomas Grießbach | **DEV**
griessbach@portrino.de

Zsuzsa Rötter | **CEO**
roetter@portrino.de



www.portrino.de