Die **config.yaml**
das unbekannte Wesen

Seit 2010 mit Contao

2010

localconfig.php

Eine Datei für alles: Datenbank-Zugang, Encryption Key, Install-Passwort und jede Einstellung aus dem Backend.

2015

parameters.yml

Contao 4 baut auf Symfony. Zugangsdaten und Secret ziehen aus, die localconfig behält nur die Einstellungen.

heute

.env + config.yaml

Secrets bleiben auf dem Server, die Konfiguration kommt ins Repository. Versionierbar und pro Umgebung steuerbar.

Zur Erinnerung: localconfig.php anno 2010

```
### INSTALL SCRIPT START ###
$GLOBALS['TL_CONFIG']['licenseAccepted'] = true;
$GLOBALS['TL_CONFIG']['installPassword'] = '9f3a…:c71e…';
$GLOBALS['TL_CONFIG']['encryptionKey'] = 'd41d8cd98f00b204e9800998ecf8427e';
$GLOBALS['TL_CONFIG']['dbHost'] = 'localhost';
$GLOBALS['TL_CONFIG']['dbUser'] = 'web123';
$GLOBALS['TL_CONFIG']['dbPass'] = 'geheim123';
$GLOBALS['TL_CONFIG']['dbDatabase'] = 'usr_web123_1';
$GLOBALS['TL_CONFIG']['websiteTitle'] = 'Meine erste Contao-Seite';
$GLOBALS['TL_CONFIG']['adminEmail'] = 'info@example.com';
$GLOBALS['TL_CONFIG']['dateFormat'] = 'd.m.Y';
### INSTALL SCRIPT STOP ###
```

Damals wie heute: klicken oder Master-DB

WEG 1

Zusammenklicken

Jede Installation von Hand im Backend einrichten.

↻ **Änderung? Überall erneut durchklicken.**

WEG 2

Master-DB einspielen

Vorlage-Datenbank importieren, alles ist schon drin.

↻ **Änderung? Hochfahren, anpassen, neu exportieren.**

GENERVT, ALSO NACHGEFORSCHT

config.yaml

einmal aufschreiben

versionieren

in jede Installation mitnehmen

↻ **Änderung? Datei ändern, deployen,** `cache:clear`**.**
Nur in Prod, in Dev automatisch.

Anatomie einer config.yaml

01 **Root-Key** sagt Symfony, welches Bundle gemeint ist.

02 **Ebenen** bilden den Konfigurationsbaum.

03 **Schlüssel: Wert** steht am Ende eines Zweigs.

04 **Einrückung** legt die Ebene fest. Nur Leerzeichen, nie Tabs.

Sascha Müller · Contao Summit 2026

```yaml
# Root-Key: welches Bundle?
contao:
  backend:
    badge_title: dev
  security:
    two_factor:
      enforce_backend: true
```

```yaml
contao:
  # Ebenen bilden den Baum
  backend:
    badge_title: dev
  security:
    two_factor:
      enforce_backend: true
```

```yaml
contao:
  backend:
    # Schlüssel: Wert
    badge_title: dev
  security:
    two_factor:
      enforce_backend: true
```

```yaml
# Einrückung = Ebene. Nur Leerzeichen!
contao:
··backend:
····badge_title: dev
··security:
····two_factor:
······enforce_backend: true
```

Contao ist installiert, und die config.yaml?

Gibt es noch nicht. Contao läuft mit seinen Standardwerten.

Eine eigene `config/config.yaml` legt man an, wenn man etwas ergänzen oder überschreiben will.

```
projekt/
└── .env.local      # DB + Secret
```

```
# .env.local
DATABASE_URL="mysql://db:db@db:3306/db"
APP_SECRET=8b211ee71d998ff76124302bd…
```

Die erste eigene config.yaml

01 **config/config.yaml anlegen** Endung `.yaml`, nicht `.yml`. Das ist veraltet und geht ab Contao 6 nicht mehr.

02 **Admin-E-Mail** kommt per `%env()%` aus der `.env.local`.

03 **SMTP** braucht nur `MAILER_DSN` in der `.env.local`.

```yaml
# config/config.yaml
contao:
  localconfig:
    adminEmail: '%env(ADMIN_EMAIL)%'
```

```
# .env.local
DATABASE_URL="mysql://db:db@db:3306/db"
APP_SECRET=8b211ee71d998ff76124302bd…
ADMIN_EMAIL=email-admin@domain.tld
MAILER_DSN=smtp://versand%40kunde.de:geheim@kunde.de:587
```

Exkurs: Postfach nur für den Versand

Gevibecodetes Formular
auf einer anderen Seite im selben Hosting

→

Server kompromittiert
Angreifer kann Dateien lesen

→

.env.local gelesen
SMTP-Passwort im Klartext

→

Postfach offen
Login mit den echten Zugangsdaten

VERSAND ÜBER INFO@

Datenschutzbruch

Kundenanfragen und Korrespondenz sind abrufbar. Meldepflicht.

VERSAND ÜBER VERSAND@

Ärgerlich, mehr nicht

Schlimmstenfalls Spam. Passwort ändern, keine Daten weg.

Mehrere Websites, mehrere Postfächer

01 **MAILER_DSN** reicht für einen Server.

02 **framework.mailer** legt mehrere Server an, Zugangsdaten aus der `.env.local`.

03 **contao.mailer** macht sie im Backend auswählbar, mit lesbarem Namen.

04 **Achtung:** Mit eigenen Transports wird `MAILER_DSN` ignoriert.


```
# .env.local
MAILER_DSN=smtp://user:pass@smtp.example.com:587
```

```
# .env.local
WEBSITE_SMTP=smtp://user:pass@smtp.website1.de:587
WEBSITE_SMTP2=smtps://user:pass@smtp.website2.de:465
```

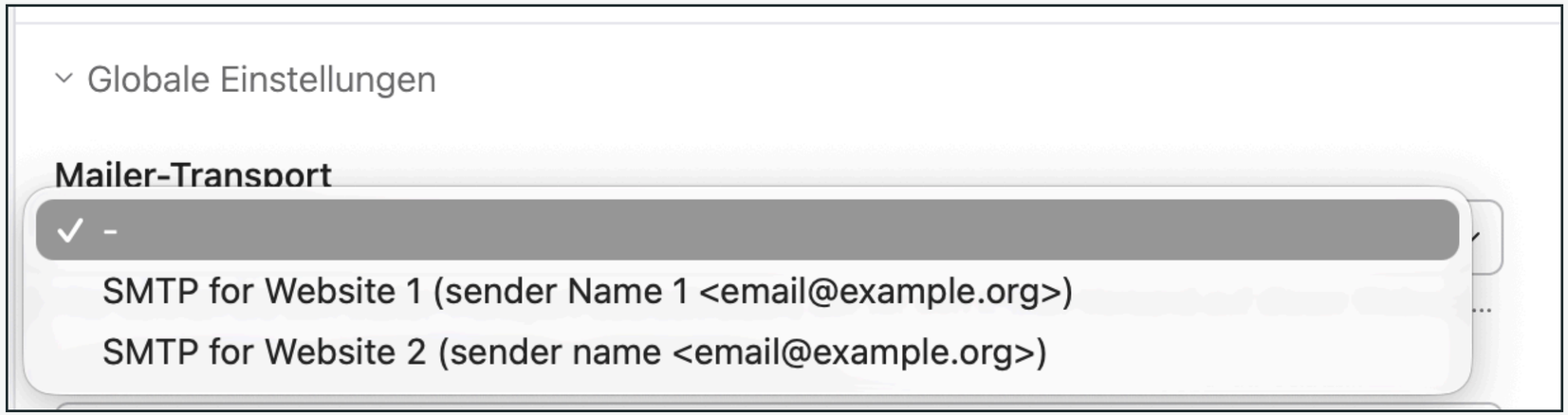
```
# config/config.yaml
contao:
  mailer:
    transports:
      website1: ~
      website2:
        from: versand@website2.de

framework:
  mailer:
    transports:
      website1: '%env(WEBSITE_SMTP)%'
      website2: '%env(WEBSITE_SMTP2)%'
```

```
# translations/mailer_transports.de.yaml
website1: 'Versand Website 1'
website2: 'Versand Website 2'
```

Mailer-Transport am Startpunkt

Name aus der Übersetzung, in Klammern der Absender aus `from`.
Ohne `from` steht nur der Name da.



Bildgrößen einmal definiert

01 **contao.image.sizes** sammelt alle eigenen Bildgrößen.

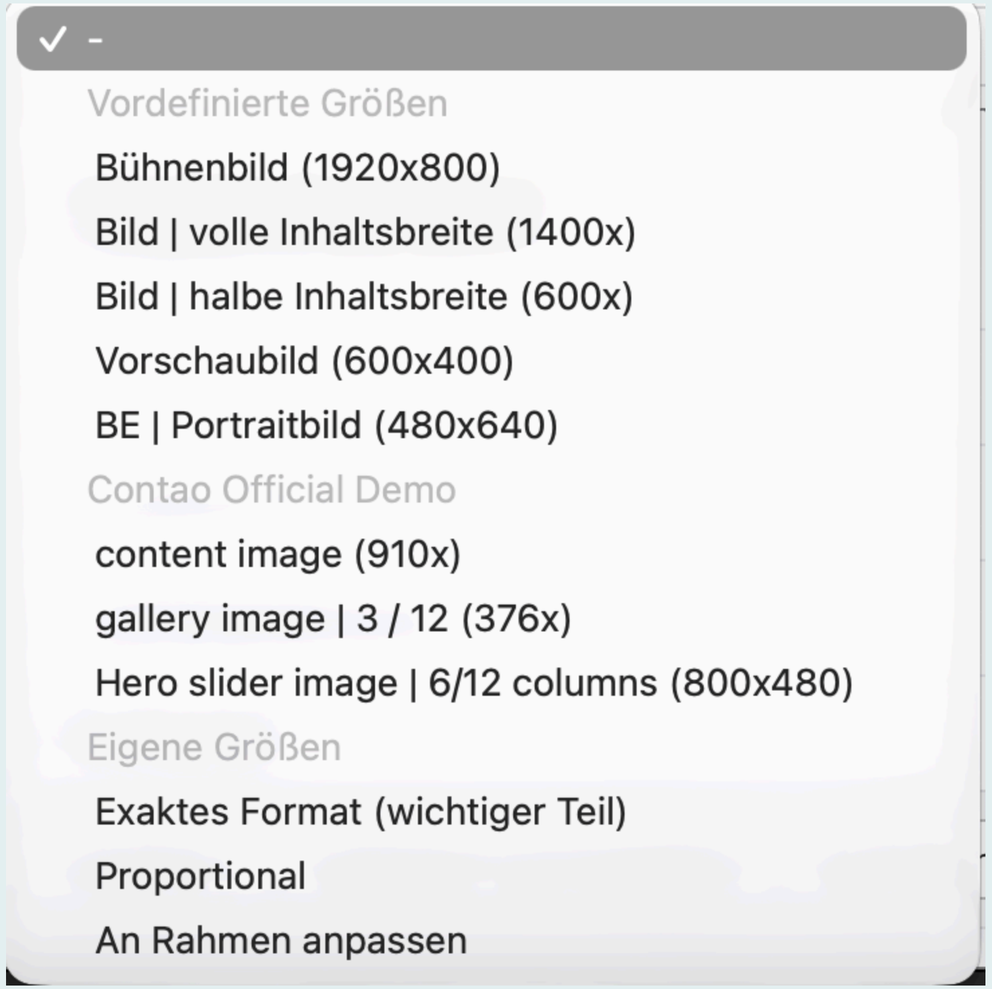
02 **_defaults** gelten für alle Größen: WebP, Lazy Loading, Retina.

03 **Namen** nur Kleinbuchstaben, Ziffern, Unterstrich. `be_` markiert bei mir Größen, die bei der Rechtevergabe nicht freigegeben werden.


```yaml
# config/config.yaml
contao:
  image:
    imagine_options:
      jpeg_quality: 90
    sizes:
      _defaults:
        lazy_loading: true
        formats:
          jpg: [ webp, jpg ]
          jpeg: [ webp, jpeg ]
          png: [ webp, png ]
        densities: 1.5x,2x
```

```yaml
# config/config.yaml
contao:
  image:
    sizes:
      hero_image:
        width: 1920
        height: 800
        resize_mode: crop
        items:
          - media: '(max-width: 767px)'
            width: 768
            height: 576
            resize_mode: proportional
          - media: '(max-width: 1199px)'
            width: 1200
            height: 600
            resize_mode: proportional
```

```yaml
# config/config.yaml
contao:
  image:
    sizes:
      hero_image: # …
      be_portrait:
        width: 480
        height: 640
        resize_mode: crop
```



Lesbare Namen im Backend

Ohne Übersetzung steht in der Bildauswahl der technische Name. Mit einer Zeile pro Größe sieht die Redaktion, wofür sie gedacht ist.

```yaml
# translations/image_sizes.de.yaml
hero_image: Bühnenbild
content_full_size: Bild | volle Inhaltsbrei
content_half: Bild | halbe Inhaltsbreite
teaser: Vorschaubild
be_portrait: BE | Portraitbild
```

localconfig in der config.yaml

01 **contao.localconfig** nimmt alles auf, was sonst unter System → Einstellungen steht.

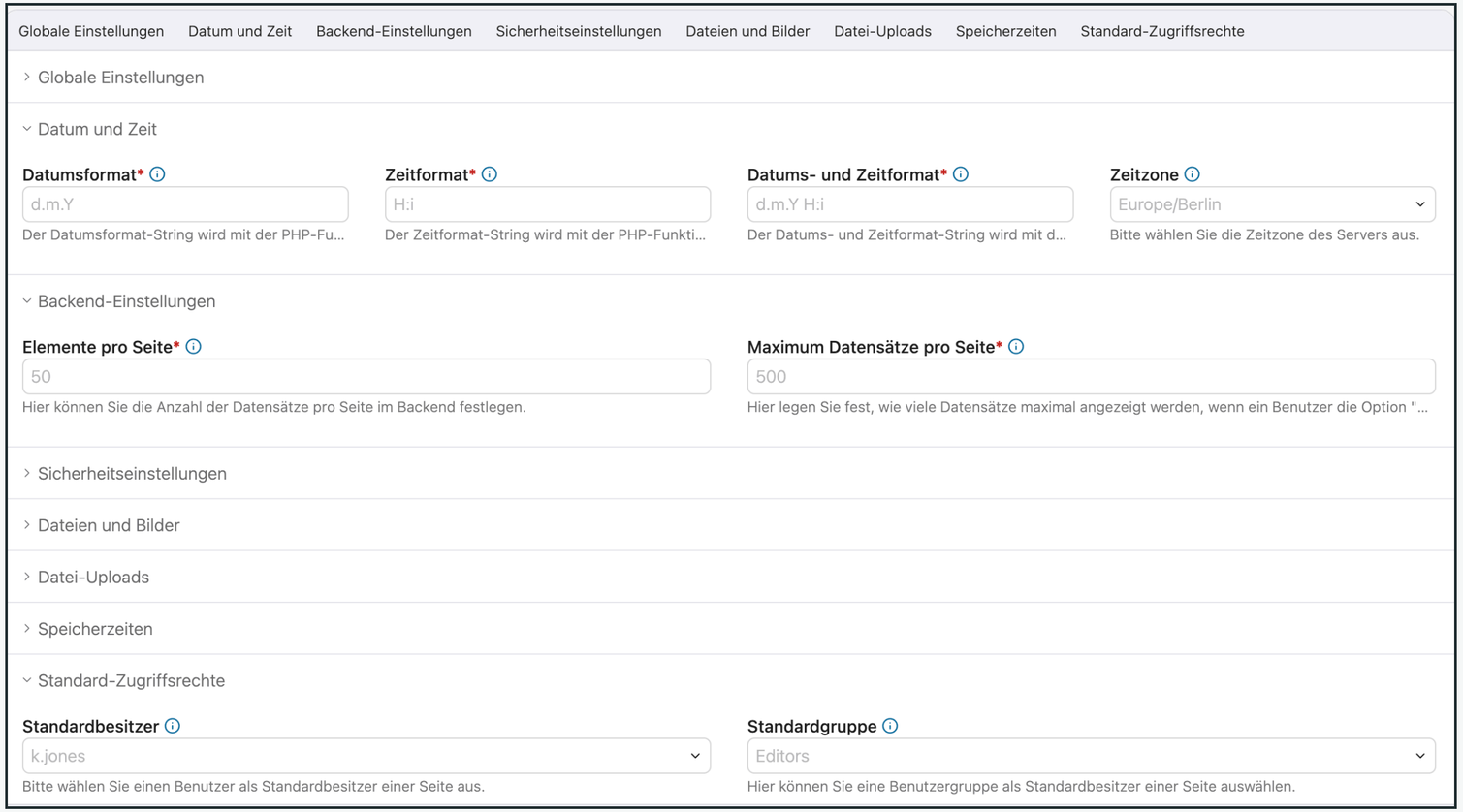
02 **Schlüssel** sind die alten Namen aus der `localconfig.php`.

03 **Gesperrt:** Was hier steht, lässt sich im Backend nicht mehr ändern.

```yaml
# config/config.yaml
contao:
  localconfig:
    adminEmail: '%env(ADMIN_EMAIL)%'
    dateFormat: d.m.Y
    datimFormat: d.m.Y H:i
    timeFormat: H:i
    timeZone: Europe/Berlin
    resultsPerPage: 50
    maxResultsPerPage: 500
    defaultUser: 1
    defaultGroup: 1
```

```yaml
# config/config.yaml
contao:
  localconfig:
    # …
    maxFileSize: 209713400 # 200 MB
    imageWidth: 5000
    imageHeight: 5000
    uploadTypes: jpg,jpeg,png,svg,pdf,zip
    undoPeriod: 5184000    # 60 Tage
    versionPeriod: 7776000 # 90 Tage
    logPeriod: 604800      # 7 Tage
```

Gesperrt im Backend



Kleine Helfer unter contao:

Option	Standard	Wofür
`registration.expiration`	14 Tage	Unbestätigte Registrierungen aufräumen
`pagination.default_range`	7	Wie viele Seitenzahlen die Pagination zeigt
`sanitizer.allowed_url_protocols`	http, https, mailto, tel …	Eigene Link-Protokolle wie `customprotocol:` erlauben
`altcha`	SHA-256 · 100000 · 24 h	Schwierigkeit und Laufzeit des Spamschutzes

Backups mit Plan

01 **ignore_tables** lässt Tabellen weg, die sich nie ändern.

02 **keep_max** legt fest, wie viele Backups höchstens bleiben.

03 **Achtung:** Wer eine eigene Tabelle ergänzt, muss die fünf Contao-Tabellen mit aufschreiben. Sonst landen sie wieder im Backup.

```yaml
# config/config.yaml
contao:
  backup:
    ignore_tables:
      - tl_crawl_queue
      - tl_log
      - tl_search
      - tl_search_index
      - tl_search_term
      # + eigene statische Tabellen
```

```yaml
# config/config.yaml
contao:
  backup:
    ignore_tables:
      # …
    keep_max: 3  # Standard: 5
```

DNS-Mapping pro Umgebung

01 **Live-Datenbank lokal einspielen:** Die Startpunkte zeigen noch auf die Live-Domain.

02 **DNS_MAPPING** in der `.env.local` sagt, welche Domain wohin soll.

03 **contao:migrate** schreibt die Domains um, dauerhaft in der Datenbank.

04 **Live umgekehrt:** lokale Domain wird wieder zur Live-Domain.

```
# .env.local (lokal)
# "Domain im Startpunkt": "neue Domain"
DNS_MAPPING='{
  "www.kunde.de": "kunde.ddev.site",
  "www.kunde.at": "kunde-at.ddev.site"
}'
```

```
# Live-DB eingespielt, dann:
$ vendor/bin/contao-console contao:migrate

# tl_page: dns = kunde.ddev.site
```

Badge fürs Backend

01 **badge_title** zeigt die Umgebung oben im Backend und beim Login.

02 **Text pro Server** kommt aus der `.env.local`.

03 **Live leer lassen,** dann gibt es keinen Badge. Fehlen darf die Variable nicht.

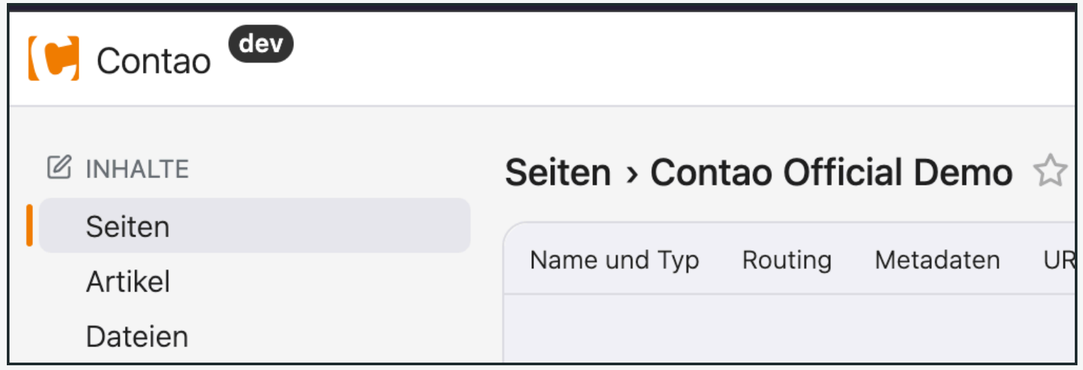
```yaml
# config/config.yaml
contao:
  backend:
    badge_title: '%env(BADGE_TITLE)%'
```

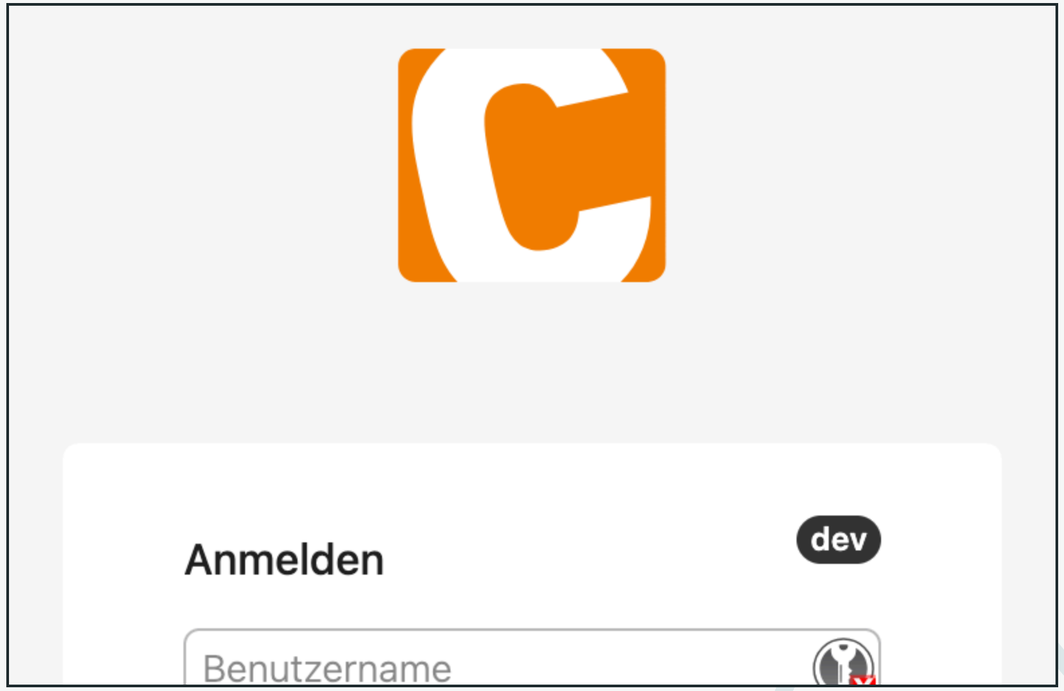
```
# .env.local lokal
BADGE_TITLE=dev

# .env.local Stage
BADGE_TITLE=stage

# .env.local live
BADGE_TITLE=     # kein Badge, oder: live
```

Badge ohne eigenes CSS





Badge gestalten

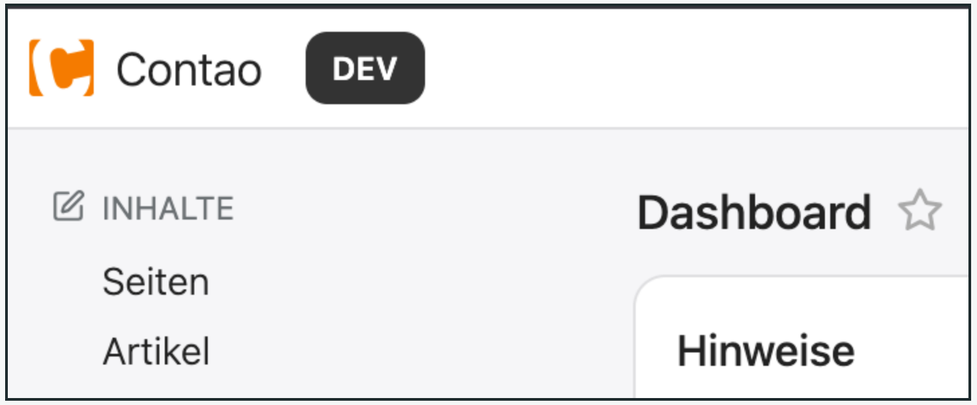
01 **attributes** hängt Klassen als `data-app-name` an den `<body>`.

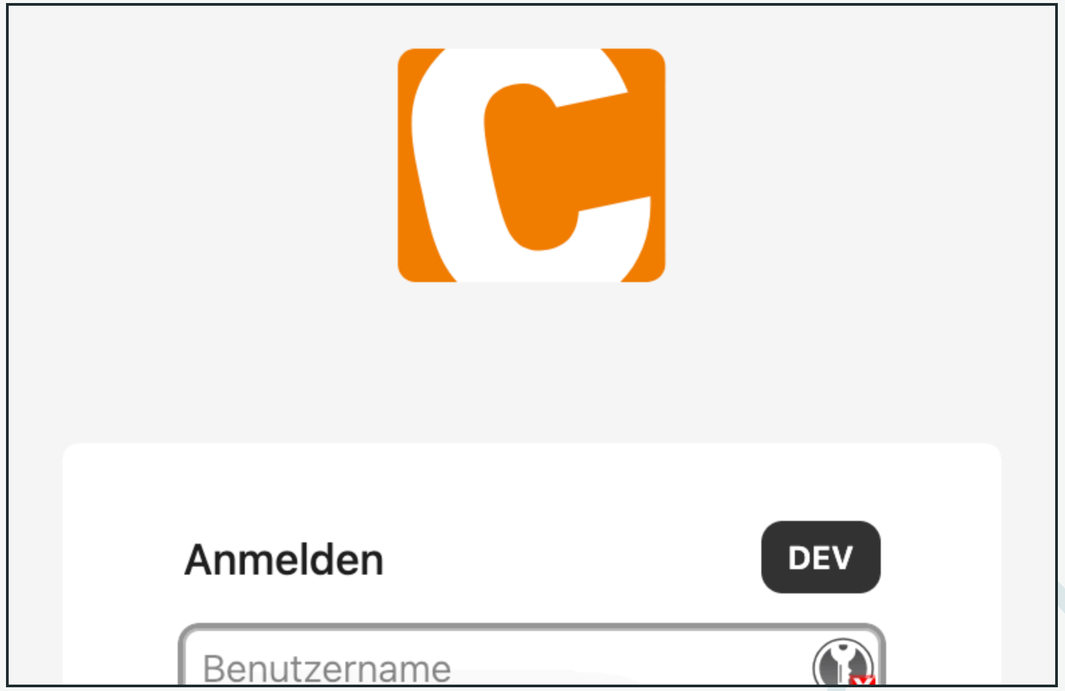
02 **custom_css** bindet eigenes CSS ins Backend ein.

03 **BE_STATUS** in der `.env.local` steuert die Farbe.

```yaml
# config/config.yaml
contao:
  backend:
    attributes:
      app-name: 'be-styles be-status--%env(BE_STATUS)%'
    custom_css:
      - files/layout/css/backend.css
    badge_title: '%env(BADGE_TITLE)%'
```

Badge mit eigenem CSS





Tipp: Assets mit Manifest

01 **Build-Tool** schreibt Dateinamen mit Hash in eine `manifest.json`.

02 **json_manifest_path** sagt Symfony, wo die Datei liegt.

03 **Im Template** reicht der feste Name, Symfony setzt den Hash ein.

```yaml
# config/config.yaml
framework:
  assets:
    json_manifest_path:
      '%kernel.project_dir%/public/layout/manifest.json'
```

```json
// public/layout/manifest.json
{
  "app.css": "/layout/app.3f9a1c.css",
  "app.js": "/layout/app.81bd04.js"
}
```

```twig
{# Template #}
<link rel="stylesheet"
      href="{{ asset('app.css') }}">

{# Ausgabe #}
<link rel="stylesheet"
      href="/layout/app.3f9a1c.css">
```

Werkzeuge auf der Konsole

01 **dump-reference** zeigt alle Optionen der installierten Version.

02 **debug:config** zeigt, was am Ende wirklich gilt.

03 **mailer:test** schickt eine Testmail, auch über einen bestimmten Transport.

04 **cache:clear** nach jeder Änderung in Prod.

```
# alle Optionen (lesbarer: Doku)
$ contao-console config:dump-reference contao

# was am Ende wirklich gilt
$ contao-console debug:config contao

# Testmail über einen Transport
$ contao-console mailer:test du@example.com \
    --transport=website1

# nach Änderungen in Prod
$ contao-console cache:clear --env=prod
```

Dev und Prod getrennt

01 **Eine Datei pro Umgebung:** Contao lädt `config_<APP_ENV>.yaml` statt der `config.yaml`.

02 **error_level** lässt sich pro Umgebung anpassen.

03 **config_prod.yaml** ist optional. Ohne sie gilt live einfach die `config.yaml`.

```yaml
# config/config_dev.yaml
contao:
  error_level: 8181  # ohne Warnungen

parameters:
  contao.dns_mapping:
    www.kunde.de: kunde.ddev.site
```

```yaml
# config/config_prod.yaml
contao:
  security:
    two_factor:
      enforce_backend: true

parameters:
  contao.dns_mapping:
    kunde.ddev.site: www.kunde.de
```

Alles in einer config.yaml

```yaml
# config/config.yaml
contao:
  backend:
    badge_title: '%env(BADGE_TITLE)%'
  localconfig:
    adminEmail: '%env(ADMIN_EMAIL)%'
    dateFormat: d.m.Y
    datimFormat: d.m.Y H:i
    timeFormat: H:i
    timeZone: Europe/Berlin
    resultsPerPage: 50
    maxResultsPerPage: 500
    defaultUser: 1
    defaultGroup: 1
    maxFileSize: 209713400
    imageWidth: 5000
    imageHeight: 5000
    uploadTypes: jpg,jpeg,png,svg,pdf,zip
    undoPeriod: 5184000
    versionPeriod: 7776000
    logPeriod: 604800
  image:
    imagine_options:
      jpeg_quality: 90
    sizes:
      _defaults:
        lazy_loading: true
        formats:
          jpg: [ webp, jpg ]
          jpeg: [ webp, jpeg ]
          png: [ webp, png ]
        densities: 1.5x,2x
      hero_image:
        width: 1920
        height: 800
        resize_mode: crop
        items:
          - media: '(max-width: 767px)'
            width: 768
            height: 576
            resize_mode: proportional
          - media: '(max-width: 1199px)'
            width: 1200
            height: 600
            resize_mode: proportional
      be_portrait:
        width: 480
        height: 640
        resize_mode: crop
  backup:
    ignore_tables:
      - tl_crawl_queue
      - tl_log
      - tl_search
      - tl_search_index
      - tl_search_term
    keep_max: 3
  mailer:
    transports:
      website1: ~
      website2: ~

framework:
  mailer:
    transports:
      website1: '%env(WEBSITE_SMTP)%'
      website2: '%env(WEBSITE_SMTP2)%'
  assets:
    json_manifest_path: '…/layout/manifest.json'
```

imports verbinden Dateien

01 **Thematische Dateien** halten die config.yaml kurz.

02 **Preset** für das nächste Projekt: Datei kopieren, fertig.

03 **Umgebungsdateien** importieren die `config.yaml`, sonst fehlt dort alles.

```yaml
# config/config.yaml
imports:
  - { resource: config_images.yaml }
  - { resource: config_localconfig.yaml }

contao:
  backend:
    badge_title: '%env(BADGE_TITLE)%'
  # …
```

```yaml
# config/config_dev.yaml
imports:
  - { resource: config.yaml }

contao:
  error_level: 8181  # ohne Warnungen
```

Warnungen ins eigene Log

01 **packages/monolog.yaml** kommt per `imports` dazu.

02 **Handler** schreibt jede Warnung in eine eigene Datei.

03 **Live** sieht der Besucher nichts, im Log steht trotzdem alles.

```yaml
# config/config.yaml
imports:
  - { resource: packages/monolog.yaml }
```

```yaml
# config/packages/monolog.yaml
monolog:
  handlers:
    php_warnings:
      type: stream
      path: '%kernel.logs_dir%/php_warnings.log'
      level: warning
```

Zum Nachlesen

Einstellungen per config.yaml

docs.contao.org/5.x/manual/en/system/settings/#config-yaml

MAILER_DSN-Konverter

docs.contao.org/5.x/manual/en/system/settings/#convert-your-mail-parameters

Konfigurationsreferenz

docs.contao.org/5.x/dev/reference/config/

PHP Error Reporting Calculator

maximivanov.github.io/php-error-reporting-calculator/

PHP-Warnungen speichern

erdmann-freunde.de/wissen/wartung/updates/php-warnungen-speichern

Fragen? Gerne.