Backend Routes :: Contao Deve

docs.contao.org/dev/guides/backend-routes/

Contao

Developer Documentation

Search...

- Getting Started ✓
- Framework ✓
- Reference ✓
- Guides ✓
  - Backend Routes ✓
  - Managing Data Records
  - Namespaces & Class Names
  - Publishing Bundles
  - Your First Extension
- Internals

Clear History

Contao Developer Documentation > Guides > Backend Routes

Adding custom backend routes

You can use the Contao backend to display content generated in your own custom Controllers. This way you can develop custom extensions without the need to use DCA configuration. The following example can be changed according to your own setup. For example you're not obliged to use the annotation configuration for your routes you could use XML or YAML interchangeably.

Create your Controller and Template

The first step is to create your own Controller. A more detailed explanation on how Symfony Controller work can be found in the Symfony documentation. The Controller class is placed inside the `Controller` directory and is configured through annotations.

```
// src/Controller/BackendController.php
namespace App\Controller;

use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Template;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;

/**
 * @Route("/contao", defaults={
```

Publishing Bundles :: Contao D
docs.contao.org/dev/guides/publishing-bundles/
Contao
Developer Documentation
Search...
Getting Started
Framework
Reference
Guides
Backend Routes
Managing Data Records
Namespaces & Class Names
Publishing Bundles
Your First Extension
Internals
Clear History

Contao Developer Documentation > Guides > Publishing Bundles

Private and commercial packages

If users need to pay for your package or you created it just for a single client, you might not want to publish it on packagist.org. There are still various ways to install it with Composer, for example by following what Private Packagist has to offer. Unfortunately, bypassing packagist.org means users will have to manually adjust their `composer.json` to add repositories and requirements, which is cumbersome and error-prone for non-developers.

Composer has always allowed to install packages from ZIP archives, which are called *Artifacts*. Using this feature, the Contao Manager allows users to upload ZIP archives and install packages without a need to manually adjust the `composer.json`. We are distinguishing between two types of *artifacts* in the Contao Manager.

1. a *regular* artifact is any Composer package packed into a ZIP archive.
2. a `contao-provider` is a special type of artifact, which adds private repositories to require additional packages into a Contao installation.

Creating an Artifact Package

Creating an artifact package for the Contao Manager is no difference from a regular Composer artifact. Its a ZIP archive of all package files, including a `composer.json` in the root of the ZIP archive.
There is one significant additional requirement: You **must** add a `version` property to your `composer.json`.