

Starting your Development :: C x

+

←

→

↺


docs.contao.org/dev/getting-started/starting-development/

☆


!

✓

🔥



⋮



Contao

Developer Documentation

🔍 Search...

✕

Getting Started ✓

Initial Setup ✓

Core Concepts ✓

Starting your Development ✓

Adjusting the DCA

Changing Translations

Implementing Hooks

Content Elements & Modules

Creating an Extension

Framework

Reference

Guides

Internals

☰

[Contao Developer Documentation](#) > [Getting Started](#) > Starting y...


# Structure


After a fresh install of Contao, your project will have a certain initial file & directory structure (which is similar to the structure of a pure Symfony project using the `symfony/skeleton` for example).

File/Directory	Explanation
<code>assets/</code>	JavaScript and CSS assets of the Contao framework and third parties.
<code>config/</code>	Application configuration files.
<code>files/</code>	Public or protected files managed by Contao's file manager.
<code>system/</code>	Legacy folder for Contao 3 compatibility.
<code>templates/</code>	Customized Contao & Twig templates.
<code>var/</code>	Transient files like the application cache and log files.
<code>vendor/</code>	Composer's vendor folder containing all dependencies (including Contao).
<code>web/</code>	Public entry points; contains symlinks to other public resources.
<code>composer.json</code>	<code>composer.json</code> of your project defining your dependencies and autoloading.

Creating an Extension :: Contao

docs.contao.org/dev/getting-started/extension/

☆ ⓘ 🔒 🔥 | 

Contao

Developer Documentation

Search...

Getting Started

Initial Setup

Core Concepts

Starting your Development

Adjusting the DCA

Changing Translations

Implementing Hooks

Content Elements & Modules

Creating an Extension

Framework

Reference

Guides

Internals

Contao Developer Documentation > Getting Started > Creating ...

# Creating the Bundle

Now it is time to do some ground work for the extension:

1. Create a bundle class.
2. Create a Contao Manager Plugin to load the bundle within a Contao Managed Edition.
3. Configure the `composer.json` for the Contao Manager Plugin.

Creating the bundle class is simple enough. The name of the bundle class can be freely choosen - typically it will have the same name as your top-level subnamespace, or even a combination of your complete top-level namespace. For example:

```
// src/ContaoExampleBundle.php
namespace SomeVendor\ContaoExampleBundle;

use Symfony\Component\HttpKernel\Bundle\Bundle;

class ContaoExampleBundle extends Bundle
{
}
```

The bundle class can be empty, but could contain additional bundle configurations (see