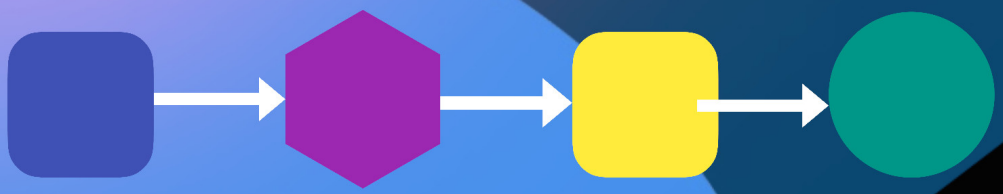
Drupal 10+Theming

How to convert an HTML template to a Drupal theme



Raynald Mompoint

Drupal 10+ Theming - How to convert an HTML template to a Drupal theme

Raynald Mompoint

Preface

Drupal is a popular open-source content management system (CMS) that allows individuals and organizations to create and manage websites, web applications and digital content. At the time of this writing, Drupal 10 (released in Dec 2022) is the latest major release of Drupal. It builds on Drupal 8 and Drupal 9, offering new features, improvements, and updates that enhances the CMS's functionality, performance, and user experience.

Note: At this eBook publication date (Sept 2024), Drupal 11 has been released (Aug 1 2024). This eBook uses Drupal 10.1 but the material and information contained within is applicable to any Drupal 10.x or Drupal 11 site.

In *Drupal 10+ Theming - How to convert an HTML template to a Drupal Theme*, we focus on Drupal 10 theming. Starting with an HTML template and a vanilla Drupal 10 site, we'll go step-by-step through the process of creating a Drupal 10 theme from scratch. Most theming courses and tutorials theme a basic site, in this eBook we'll be theming a fairly complex site.

In the following sections we'll use "Drupal" generically to refer to Drupal versions 8+. We'll use Drupal 10 or the abbreviation "D10", if a particular topic or feature only applies to Drupal 10. We won't cover topics or features unique to Drupal 11.

Who this eBook is for

Drupal 10+ Theming - How to convert an HTML template to a Drupal Theme is mostly intended for website developers involved with Drupal but also for anyone interested in Drupal theming.

It assumes knowledge of HTML, CSS and familiarity with the Twig templating language. Basic knowledge of PHP is helpful as we'll be writing a bit of PHP code. JavaScript knowledge is helpful but not necessary as we won't be writing any custom JavaScript. The HTML template uses Bootstrap 5, therefore Bootstrap 5 knowledge is advantageous.

To get the most of out of this book, prior experience with Drupal and basic Drupal site building is helpful.

In particular:

- You should be familiar with Drupal site administration, this includes:
    - Navigating the Drupal administration UI
    - Using composer to install and remove modules and themes
    - Enabling and configuring modules
    - Working with the theme Appearance screen to enable, disable and configure themes
    - Using the command line and Drush
- You should know how to work with Drupal Content types and Fields to create content.
- You should be familiar with Drupal core components like Blocks, Taxonomy, Views, View modes, Image styles etc. and how to configure them.
- You should have basic understanding of how Drupal configuration management works.

It would also be helpful to already have a local development environment setup on your computer. In this book we use DDEV but you're free to use your development environment of choice. Other popular development environments are Lando and MAMP-Pro.

In Chapter 2 -Setting up a Development Environment (using DDEV), we cover the basics of setting up local development environment using DDEV.

Lastly you should be familiar with Git. Git is essential knowledge for Drupal development. It also would be helpful to know how to use GitHub.

What you need for this eBook

To install the required software (DDEV local development environment, Drupal 10 codebase etc.), you'll need a computer meeting the following requirements:

- OS: macOS, Windows or Linux
- 2GHz+ Dual-core CPU
- At least 8GB RAM (16GB preferred)
- Minimum 256GB free hard drive space

In this book we use macOS but you're free to use any other OS of choice (i.e. Windows or Linux). You'll also need a text editor (or preferably) an IDE such as Visual Studio Code or PhpStorm.

What you will learn

In *Drupal 10+ Theming - How to Convert an HTML Template to a Drupal Theme*, you'll begin on a journey to master the art of Drupal theming. Through detailed, step-by-step instructions, you'll learn how to transform an HTML template into a fully functional Drupal 10 theme while building a complex business website.

- Learn how to set up a Drupal local development environment using DDEV.
- Learn how to install a local Drupal site using DDEV and Composer.
- Configure a local Drupal development environment for theme development.
- Understand the essentials of Drupal theming, including the structure of themes, theme components, and the role of Twig templates.
- Starting from scratch, build out basic theme files and set up theme directories.

- Understand the basics of Drupal configuration management. Install and use helpful configuration management related contrib modules.

- Learn how to dissect and analyze an HTML template, section by section and identify the Drupal components (Content types, Blocks, Views, Image styles etc.) needed to build and theme the section.

- Install and use site building related Drupal contrib modules such as Paragraphs, Pathauto, Webform, Block Field, Views Reference Field and others.

- Learn how to incorporate HTML template CSS, JavaScript and other asset files into a Drupal theme.

- Become proficient at theming site Headers / Footers, Hero banners, Navigation menus, Image sliders and Drupal components such as Paragraphs, Blocks, Content nodes, Views and Webforms.

- Learn how to use the browser inspector to identify the Twig template(s) being used to display site content.

- Learn how to override default Twig templates and write custom markup to style content.

- Write custom CSS and preprocess functions for a theme.

- Learn about the Twig dump() and Devel module dsm() debug commands and how to use them during theme development.

- Use the Twig Tweak and Fences Drupal contrib modules to help streamline Twig template code development.

By the end of this eBook, you will have the skills and confidence to convert any HTML template into a fully customized Drupal theme, equipping you to handle any complex Drupal theming project with ease.

What this eBook covers

Drupal 10+ Theming - How to convert an HTML template to a Drupal Theme is a practical and detailed guide for Drupal developers looking to convert HTML templates into Drupal themes. It provides step-by-step instructions, best practices, and tips to ensure a smooth theming process.

Chapter 1 - The Website We'll Be Building discusses the website (and its features) that will be built and themed throughout this eBook. The website is a business site built using the Arsha template from BootstrapMade.

Chapter 2 - Setting Up a Development Environment (using DDEV) details the process of setting up a local development environment for Drupal using DDEV. DDEV leverages Docker to create a virtual environment with a web server, PHP engine, and database server.

Chapter 3 - Install the Site focuses on the process of installing our Drupal site using DDEV and configuring the necessary settings needed for development. We also download and enable several useful Drupal contrib modules.

Chapter 4 - Configuration Management briefly discusses Drupal configuration management and introduces the Configuration Split and Config Ignore contrib modules. These two modules greatly enhance configuration management functionality.

Chapter 5 - Git Setup and Initial Commit provides guidance on setting up Git. We create a workable `.gitignore` file and perform an initial commit, We then perform an initial site database backup and DDEV environment snapshot.

Chapter 6 - Drupal 10 Theming Overview provides an overview of Drupal 10 theming, detailing the options available when selecting a theme. Theme administration and theme components (theme files, regions and blocks) are then discussed. Lastly we cover site configuration and setup needed for theme development.

Chapter 7 - Arsha Theme Prep delves into the initial steps required to prepare the Arsha theme for our Drupal site. It involves creating an initial set of theme files and setting up theme directo-

ries. We then create a theme screenshot and install the Arsha theme as the site frontend theme.

Chapter 8 - Design Mockup Review outlines the process of analyzing the Arsha Bootstrap 5 template design. The chapter begins with adding the Arsha template files to our project. We then perform a front page, section-by-section, analysis and discuss the Drupal components (and other theming considerations) we'll need for building and theming the section.

Chapter 9 - Building Out the Theme Files begins the process of building out the theme files. Building from the files created in *Chapter 7 - Arsha Theme Prep* , we update the two main theme configuration files. We also create the top level Twig template files. These latter files define the overall site page structure.

Chapter 10 - Theming the Header and Navbar focuses on building and theming the site Header, navigation bar (Navbar) and main menu links.

Chapter 11 - Building Out the Page Structure focuses on building out the basic structure of our site pages. We create a placeholder front page with dummy content. This allows us to see the overall page structure as we build and theme the various page sections. We then theme the Primary tab block.

Chapter 12 - Front Page Theming (Part 1) covers building and theming the front page Hero and Footer sections our site. Creating and configuring the section components, adding section content and styling the section is covered in detail.

Chapter 13 - Front Page Theming (Part 2) covers building and theming the front page Clients, About Us, Why Us and Frequently Asked Questions (FAQ) sections of our site. Creating and configuring the section components, adding section content and styling the section is covered in detail.

Chapter 14 - Front Page Theming (Part 3) covers building and theming the front page Skills, Services and Call to Action (CTA) sections of our site. Creating and configuring the section components, adding section content and styling the section is covered in detail. We get our first taste of working with and theming Views. We create, configure and theme of view for Services content. Theming Views is discussed in detail.

Chapter 15 - Project Page and Portfolio Section Theming covers theming the Portfolio project pages and the front page Portfolio section. Creating and configuring the project pages and Portfolio section components, adding content and styling the pages and section is covered in detail. We also get more practice working with and theming Views with the Projects view.

Chapter 16 - Front Page Theming (Part 4) covers theming the remaining front page sections, Team, Pricing and Contact . Creating and configuring the section components, adding section content and styling the section is covered in detail. We create, configure and theme views for Team and Pricing table content.

*Appendix A - Drupal Component Configuration*covers in detail creating and configuring all the Drupal components used to build the site. Screenshots and other details are provided for creating and configuring Blocks, Content types, Image styles, Paragraphs, Views and all other components.

Appendix B - Project GitHub Repo covers the site's GitHub code repo. The repo has a Git tag that "snapshots" the state of the site (files and database) at the conclusion of each eBook chapter. This allows one to "jump in" at any point in the site's development and grab the file set and database.

Appendix C - Contrib Modules provides a chapter by chapter list of Drupal contrib modules used in the site.

Appendix D - Resources provides a list of documentation resources.

Resources

- eBook website: https://drupal10plustheming.com
- GitHub repo: https://github.com/raynaldmo/Drupal-10-Plus-Theming

Chapter 1 - The Website We'll Be Building

We'll be building a business website using the Arsha template from BootstrapMade.

BootstrapMade provides free and pro versions of Bootstrap 4 / 5 templates for many types of websites. Go to Arsha - Free Corporate Bootstrap HTML Template and click the "Live Demo" button to see the Arsha template in action.

Note: This eBook uses the Arsha template available in late 2023. At eBook publication date (Sept 2024) some updates have been made to the template. Some of these changes slightly alter the look of the site. For example new content has been added and some images and icons have changed. Also the latest template version uses Bootstrap version 5.3.3 while previous versions used 5.3.2 .

A “snapshot” of the Arsha template files (from late 2023) is included in the Git repo, Drupal-10-Theming, that accompanies this eBook. The site build will be based on these files.

The Arsha template has the following main features:

- Fully responsive design using Bootstrap 5.3.2
- Single page design
- Fixed navigation menu bar
- Mobile menu
- Scroll to top button
- Portfolio section with Lightbox
- Contact form
- Animation effects

- Remix Icons

Licensing

In this book, we'll be using the free version of the Arsha template in a local D10 site for demonstration purposes. For commercial use of BootstrapMade templates, check out the Bootstrap-Made Licensing and Pricing page.

Summary

This chapter introduced the website we'll be building. The site is based the Arsha template from BootstrapMade. The Arsha template, designed with Bootstrap 5.3.2, is a single-page design, is fully responsive and features a fixed navigation bar, mobile menu, scroll-to-top button, portfolio section with Lightbox, contact form, and animation effects.

In the next chapter, we'll go through the process of setting up a local development environment for our site using DDEV.

Chapter 2 - Setting Up a Development Environment (using DDEV)

Note: *All screenshots in this chapter are available in the project Git repo at Drupal 10+ Theming Chapter 2 Screenshots.*

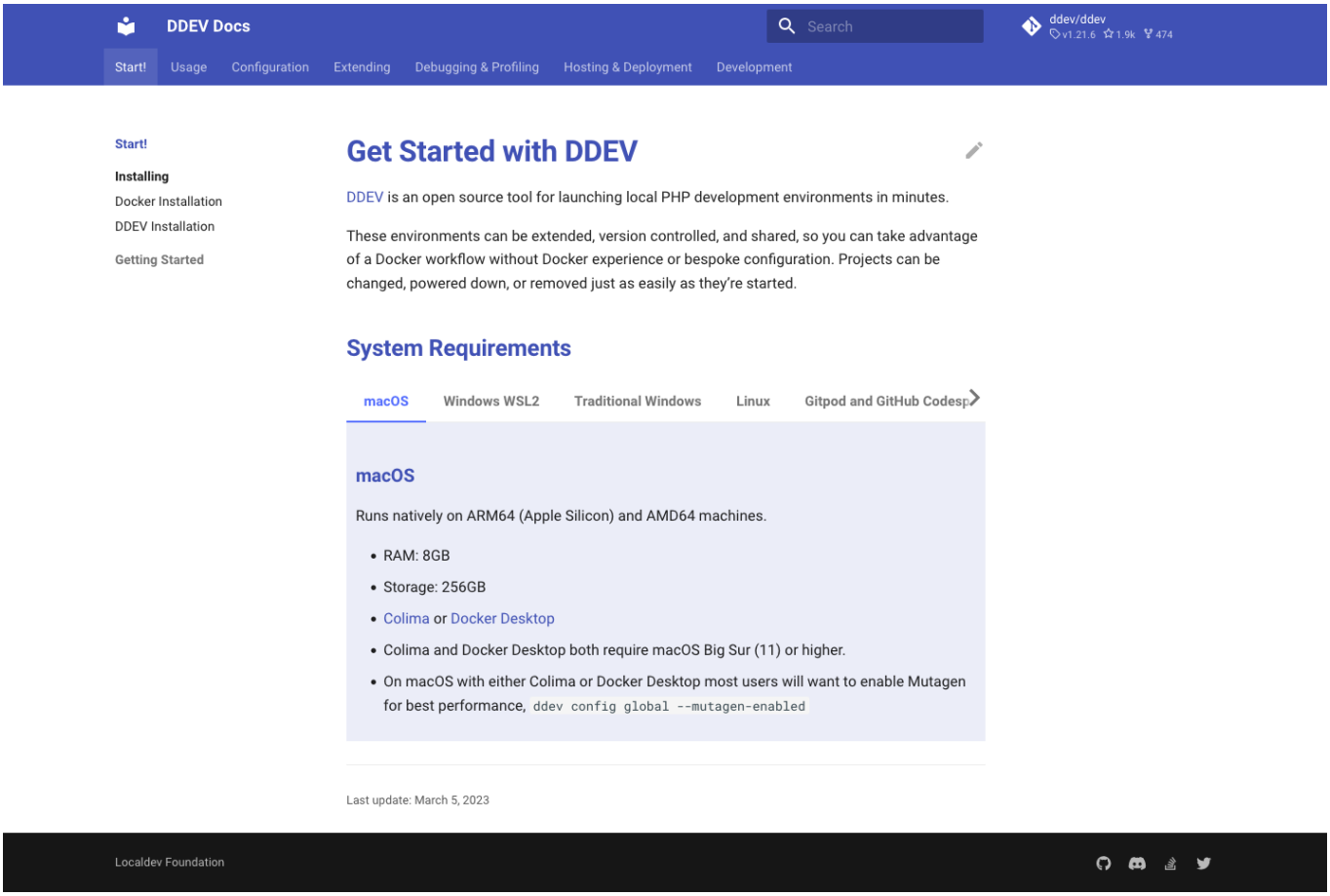
DDEV uses Docker under the hood. You must install Docker before installing DDEV. This software combination provides a virtual environment with a web server, PHP engine and database server along with other tools that are needed to run Drupal.

Note: Local Web Development with DDEV Explained is a great resource that gives background on DDEV, installation instructions and other information. (The author refers to DDEV as DDEV-Local but they are one of the same.)

In the rest of this chapter we'll refer to the official DDEV documentation to set up our local development environment.

Docker and DDEV Installation

- Navigate to Get Started with DDEV. See below screenshot.



ebook/screenshots/ch2/getting-started-with-ddev.png

Note: The DDEV Docs UI and pages are occasionally updated. The screenshots in this chapter were taken in late 2023. The DDEV Docs pages as of this eBook publication date (Sept 2024) may look different. In any case, the DDEV installation instructions should be easy to follow.

- Under "System Requirements" click on the tab for your OS and see RAM and storage requirements.

- On the left-side pane (*you may need to widen your browser window to see the pane*), the "Docker Installation" and "DDEV Installation" links will take you to installation instructions.

- In my particular case, I'm using macOS and Colima.

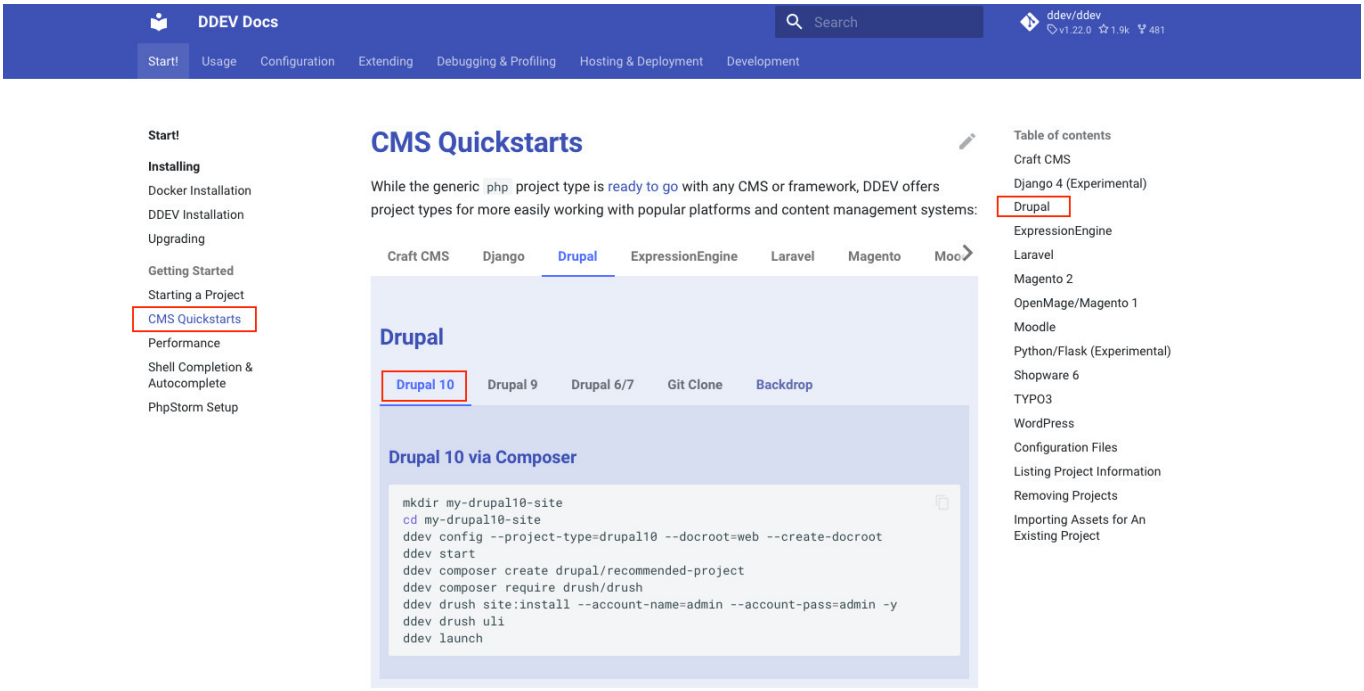
- Once you've installed Docker and DDEV, you can move on to installing Drupal.

Note: DDEV versions may differ slightly in the commands and command options they support. In this eBook we initially started with DDEV version v1.21.6 and upgraded along the way and finished with version v1.23.3. Using DDEV version v1.21.6 and up should pose no issues.

Drupal 10 Installation

From the Get Started with DDEV page in the left pane, click on the CMS Quickstarts link and then on the Drupal link on the right pane. Under Drupal 10, you'll see the set of DDEV commands needed to install a Drupal 10 site.

We won't follow these instructions verbatim and instead customize them for our project.



ebook/screenshots/ch2/cms-quick-start-d10.jpg

Use the following set of instructions to configure DDEV for Drupal 10 and install Drupal 10.

- From your home directory, create `Projects/drupal-sites/arsha` directory and cd to that directory.

```
$ pwd

/Users/<your-user-name>

$ mkdir -p Projects/drupal-sites/arsha

$ cd !$

~/Projects/drupal-sites/arsha $ pwd

/Users/<your-user-name>/Projects/drupal-sites/arsha
```

- Execute `ddev config` command

```
~/Projects/drupal-sites/arsha $ ddev config --project-type=drupal10 --
docroot=web --create-docroot --webserver-type=apache-fpm --php-ver-
sion="8.2" --database=mariadb:10.4
```

Note: Check that you're using DDEV version >= v1.21.6 otherwise the php and database versions you specify may not be supported.

We're setting the Drupal 10 source code directory to `~/Projects/drupal-sites/arsha`. If you have multiple local websites (Drupal and non-Drupal) on your system like I do, it helps to organize them in a specific directory.

We also explicitly specify web sever type (we chose Apache vs the default Nginx), php version, database engine and docroot.

After the command is run you may see the warning :

```
You have specified a project type of drupal10 but no project of that
type is found in…
```

You can safely ignore this.

You can check DDEV configuration for your site in file `.ddev/config.yaml`. Below is the first few lines of the file.

```
~/Projects/drupal-sites/arsha $ cat .ddev/config.yaml
name: arsha
type: drupal10
docroot: web
php_version: "8.2"
webserver_type: apache-fpm
router_http_port: "80"
router_https_port: "443"
xdebug_enabled: false
additional_hostnames: []
additional_fqdns: []
database:
    type: mariadb
    version: "10.4"
nfs_mount_enabled: false
mutagen_enabled: false
use_dns_when_possible: true
composer_version: "2"
web_environment: []
nodejs_version: "16"
```

- Next, run `ddev start` to start the Docker containers.

```
~/Projects/drupal-sites/arsha $ ddev start
```

```
Starting arsha...
Network ddev-arsha_default  Created
Container ddev-arsha-web  Started
Container ddev-arsha-dba  Started
Container ddev-arsha-db  Started
Starting mutagen sync process... This can take some time.
Mutagen sync flush completed in 5s.
For details on sync status 'ddev mutagen st arsha -l'
Container ddev-router  Running
Successfully started arsha
Project can be reached at https://arsha.ddev.site
https://127.0.0.1:32796
```

Note: If the `ddev start` command fails, run `ddev restart` command. (In my experience, Docker / Colima performance on MacOS isn't the best).

- Next download the Drupal 10 code with `ddev composer create` command.

```
~/Projects/drupal-sites/arsha $ ddev composer create drupal/recommended-
project:10.1.0 --no-interaction
```

We specify Drupal 10.1, *which when this eBook was started was the latest Drupal 10 version*. The site we're building should work with any Drupal 10 or Drupal 11 version but to set a base-line Drupal core version for the site, we're using version 10.1.

- Next install Drush.

```
~/Projects/drupal-sites/arsha $ ddev composer require "drush/drush:^12.0"
```

Summary

In this chapter we setup a Drupal 10 local development environment using DDEV.

- First we installed Docker and DDEV using the DDEV Docs online documentation.
- We then configured DDEV for Drupal 10, explicitly specifying the web server, PHP version, database engine and other settings.
- We then downloaded the Drupal 10 core code using composer.
- We ended by downloading Drush. Drush is an essential command-line tool for managing a Drupal site.

In the next chapter we'll move on to actually installing our Drupal site and also install some useful contrib modules.

Chapter 3 - Install the Site

Note: *All code in this chapter is available in the project Git repo at [Drupal 10+ Theming Chapter 3 Code](#) . Screenshots are available at [Drupal 10+ Theming Chapter 3 Screenshots](#).*

Before installing the site, run `ddev describe` to get the site's url, and view web server info, database info, php info as well as other details.

```
~/Projects/drupal-sites/arsha (main) $ ddev describe
```

```
Project: arsha ~/Projects/drupal-sites/arsha https://arsha.ddev.site
Docker platform: colima
Router: traefik
```

SERVICE	STAT	URL/PORT	INFO
web	OK	https://arsha.ddev.site InDocker: web:443,80,8025 Host: 127.0.0.1:32778,32779	drupal10 PHP8.2 apache-fpm docroot:'web' Perf mode: mutagen NodeJS:16
db	OK	InDocker: db:3306 Host: 127.0.0.1:32777	mariadb:10.4 User/Pass: 'db/db' or 'root/root'
Mailpit		Mailpit: https://arsha.ddev.site:8026 Launch: ddev mailpit	
All URLs		https://arsha.ddev.site, https://127.0.0.1:32778, http://arsha.ddev.site, http://127.0.0.1:32779	

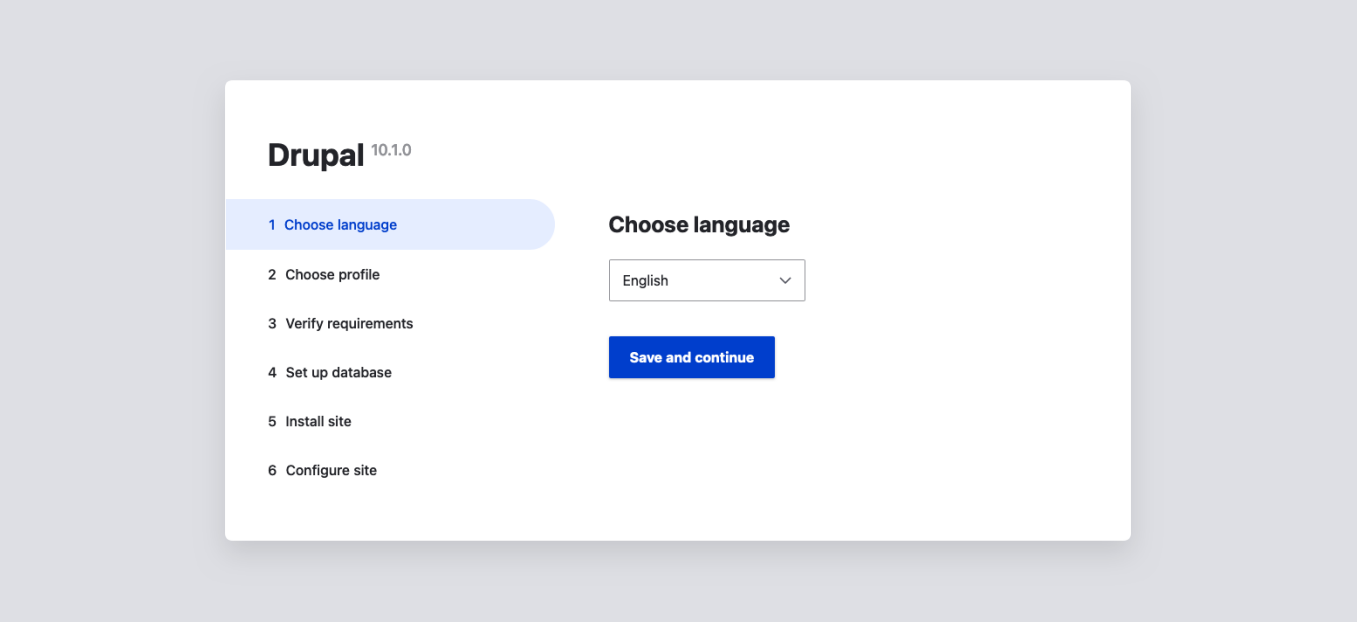
ebook/screenshots/ch3/ddev-describe.png

We see that we can access the site at url https://arsha.ddev.site or http://arsha.ddev.site. Also note a database user (`db`), and name (`db`) has been setup.

To install the site we have two options:

Option #1

Point your browser to the site's url (`https://arsha.ddev.site`) and manually go through the installation steps using the Drupal installation wizard. The first screen of the installation wizard is shown below.



ebook/screenshots/ch3/drupal-installation-wizard.png

Option #2

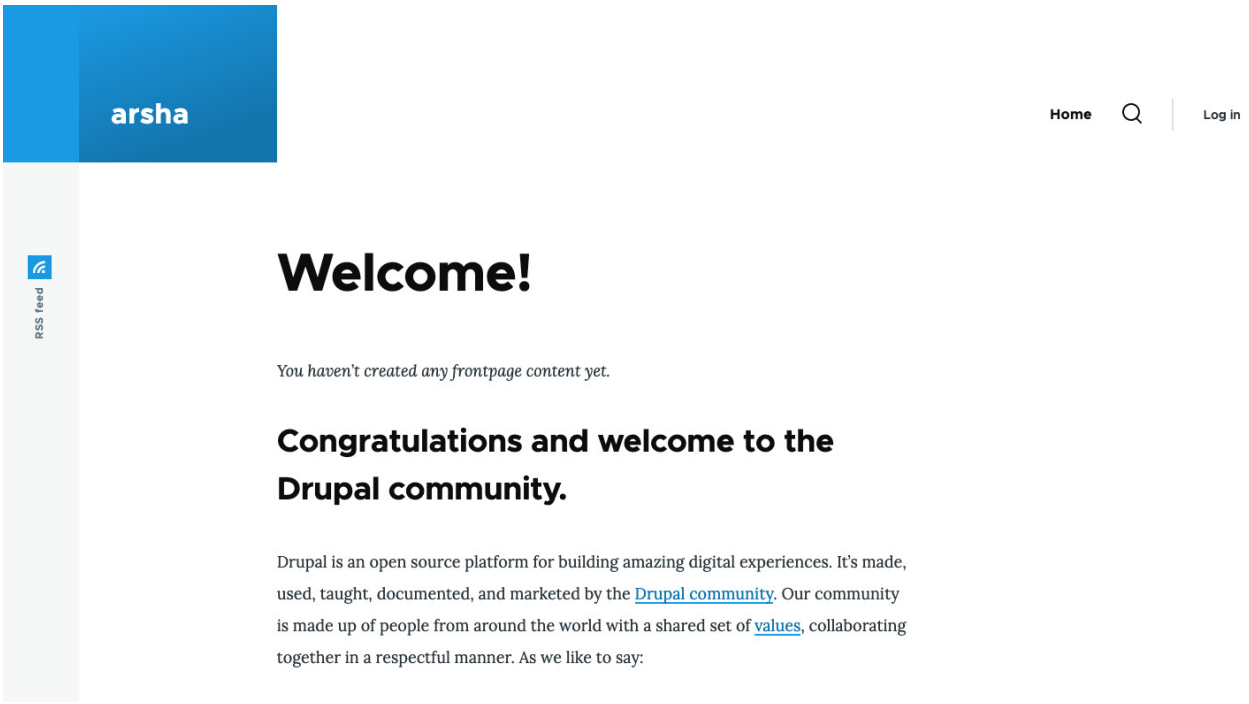
Use `ddev drush site:install` command to streamline the process and skip the manual steps. This is the method we'll use.

- From your project directory, execute the command as shown below.

```
~/Projects/drupal-sites/arsha $ ddev drush site:install -y standard --
site-name=arsha --account-name=admin --account-pass=admin

You are about to:

 * DROP all tables in your 'db' database.


 // Do you want to continue?: yes.
```

```
[notice] Starting Drupal installation. This takes a while.
[notice] Performed install task: install_select_language
[notice] Performed install task: install_select_profile
[notice] Performed install task: install_load_profile
[notice] Performed install task: install_verify_requirements
[notice] Performed install task: install_verify_database_ready
[notice] Performed install task: install_base_system
[notice] Performed install task: install_bootstrap_full
[notice] Performed install task: install_profile_modules
[notice] Performed install task: install_profile_themes
[notice] Performed install task: install_install_profile
[notice] Performed install task: install_configure_form
[notice] Performed install task: install_finished
[success] Installation complete.
```

- Navigate to https://arsha.ddev.site and voila!, your site is up and ready to go. See below screenshot. (Only the top portion of the site is shown).



ebook/screenshots/ch3/drupal-site-homepage.jpg

- Login to the site with Username: admin, Password: admin

docroot and webroot

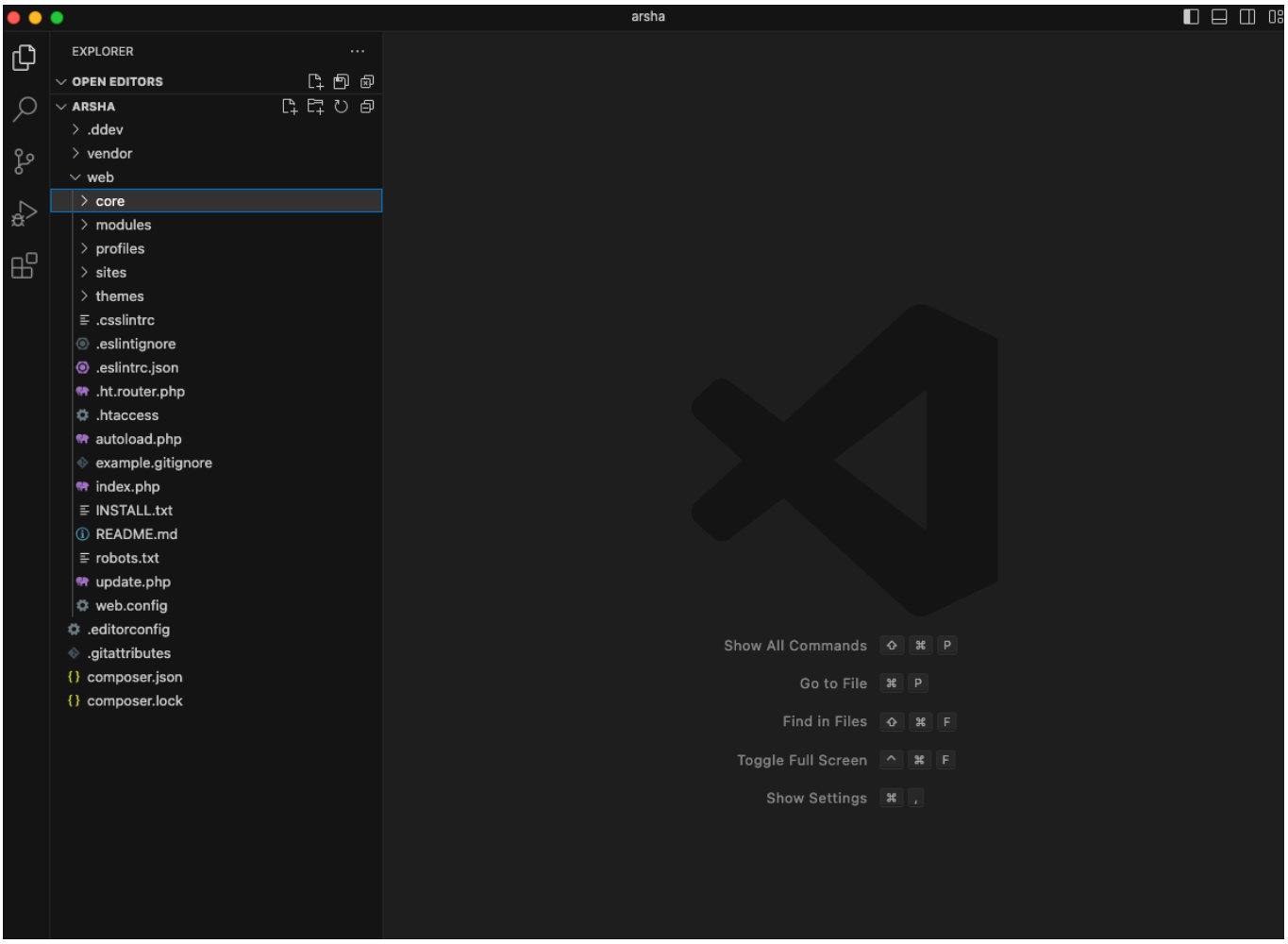
Throughout this book we may refer to the site's `docroot` and `webroot`.

`docroot` is the top level directory where the Drupal source code is located. In *Chapter 2 - Setting up a Development Environment (using DDEV) -> Drupal 10 Installation,* we installed our Drupal site at`~/Projects/drupal-sites/arsha`.

Directory `~/Projects/drupal-sites/arsha` is the site `docroot`.

The site `webroot` located at `~/Projects/drupal-sites/arsha/web`. `webroot` consists of publicly accessible files served by the web server. Directories and files located above `webroot` aren't publicly accessible and are a good place to hold private data and configuration.

The screenshot below shows the Drupal code (`docroot`) directory structure (using VScode). `webroot` files and directories are under the `web/` directory.



ebook/screenshots/ch3/drupal-code-directory-structure.png

Settings Files

Drupal has a main site settings file , `settings.php`, (located at `web/sites/default/settings.php`) that holds site specific configuration.

DDEV automatically creates a `settings.php` and a `settings.ddev.php` file. `settings.php` includes `settings.dev.php`. Both of these files have site configuration settings.

In particular, `settings.ddev.php` contains the site database settings and configuration directory setting.

Important: Open the `web/sites/default/settings.ddev.php` file. The screenshot below shows the first few lines of the file. ***Remove*** the entire`@file` annotation block at the top of the file. This "trick" ensures that any changes we make to `settings.ddev.php` are not overwritten whenever we restart the site (using the `ddev start` or `ddev restart` command).

```php
<?php

/**
 * @file
 * #ddev-generated: Automatically generated Drupal settings file.
 * ddev manages this file and may delete or overwrite the file unless this
 * comment is removed.  It is recommended that you leave this file alone.
 */

$host = "db";
$port = 3306;
$driver = "mysql";

// If DDEV_PHP_VERSION is not set but IS_DDEV_PROJECT *is*, it means we're
running (drush) on the host,
// so use the host-side bind port on docker IP
if (empty(getenv('DDEV_PHP_VERSION') && getenv('IS_DDEV_PROJECT') == 'true')) {
  $host = "127.0.0.1";
  $port = 32806;
}
```

ebook/screenshots/ch3/settings-php-ddev-php-file-annotation.png

After removing the `@file` annotation block from `settings.ddev.php`, the first few lines of the file should look as below.

```php
<?php

$host = "db";
$port = 3306;
$driver = "mysql";

// If DDEV_PHP_VERSION is not set but IS_DDEV_PROJECT *is*, it means we're
running (drush) on the host,
// so use the host-side bind port on docker IP
if (empty(getenv('DDEV_PHP_VERSION') && getenv('IS_DDEV_PROJECT') == 'true')) {
  $host = "127.0.0.1";
  $port = 32806;
}
```

ebook/screenshots/ch3/settings-php-ddev-php-no-file-annotation.png

Next, open the `settings.php` file. At the bottom of the file is code to include`settings.local.php`. If this section is commented out, un-comment it. See below.

```
if (file_exists($app_root . '/' . $site_path . '/settings.local.php')) {
   include $app_root . '/' . $site_path . '/settings.local.php';
}
```

Lastly, we need to create the `settings.local.php` as that file is not automatically created.

To create it, copy file `web/sites/example.settings.local.php` to `web/sites/default/example.settings.local.php` and rename it `settings.local.php`.

```
~/Projects/drupal-sites/arsha $ cp web/sites/example.settings.local.php
web/sites/default/settings.local.php
```

Shown below are all the files that should be in the `web/sites/default` directory.

```
~/Projects/drupal-sites/arsha$ ls web/sites/default/

default.services.ymldefault.settings.phpsettings.ddev.phpsettings.lo-
cal.phpsettings.php
```

Accessing The Site Database

DDEV includes the phpMyAdmin. Use this tool to access the site database. As previously noted, use `ddev describe` to get site information. From this output, phpMyAdmin information will be displayed.

Launch phpMyAdmin by executing `ddev launch -p` from `docroot` or by pointing your browser to `https://arsha.ddev.site:8037` .

Note: For for DDEV versions > v1.21.6 you may get the following when `ddev launch -p` is run.

```
/Projects/drupal-sites/arsha (main) $ ddev launch -p

phpMyAdmin is no longer built into DDEV, please 'ddev get ddev/ddev-php-
myadmin' and use 'ddev phpmyadmin' to launch phpMyAdmin

Failed to run launch -p; error=exit status 2
```

Follow the instructions given to launch phpMyAdmin.

Download and Install Contrib Modules

There are several Drupal contrib modules that are very useful for any type of site being built. From `docroot`, (`~/Projects/drupal-sites/arsha`), download the following modules with composer.

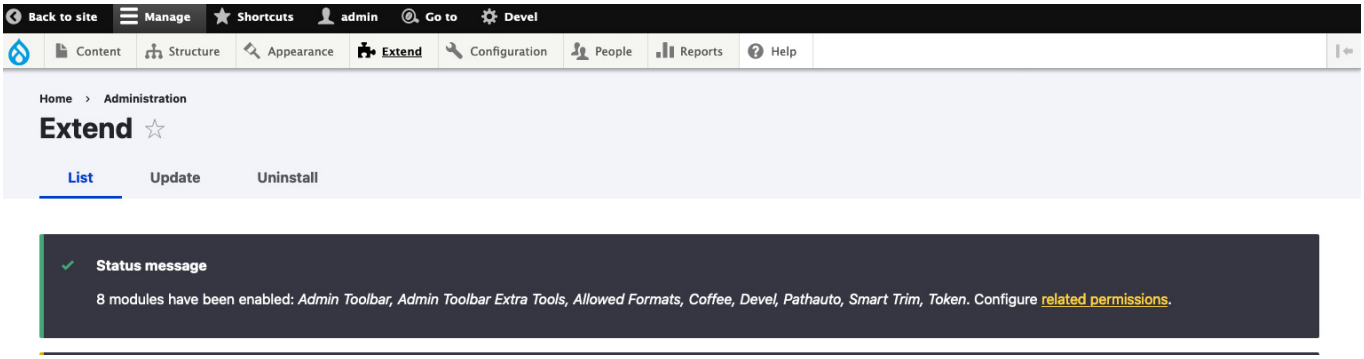
```
~/Projects/drupal-sites/arsha $ ddev composer require drupal/admin_tool-
bar

~/Projects/drupal-sites/arsha $ ddev composer require drupal/coffee

~/Projects/drupal-sites/arsha $ ddev composer require drupal/pathauto

~/Projects/drupal-sites/arsha $ ddev composer require drupal/smart_trim

~/Projects/drupal-sites/arsha $ ddev composer require drupal/allowed_for-
mats
```

```
~/Projects/drupal-sites/arsha $ ddev composer require --dev drupal/devel
```

If un-familiar with these modules, be sure to head over to the drupal.org module page to get information about the module. For example navigate to https://www.drupal.org/project/admin_toolbar for information about Admin Toolbar.

TIP: Checkout 50 Drupal modules every Drupal professional should know about for a comprehensive list of useful Drupal contrib modules. At this point in our development though, the modules listed above are adequate.

After the modules are downloaded, be sure to head over to the Extend screen (`admin/modules`) and enable the modules. The screenshot below shows a Status message after the modules have been enabled.



ebook/screenshots/ch3/initial-contrib-modules.jpg

TIP: The coffee module is one of my favorite contrib modules and is extremely useful for navigating the Drupal site admin pages. Just type `Alt-d` and enter a search term. A list of links will be shown that match the search term.

Try it now:

- Type `Alt-d` and enter “text format”
- Click on “Text formats and editors”. This will take you right to the Text formats and editors page, `admin/config/content/formats`.

This is much faster than browsing through the various admin links to find the page you want.

For the remainder of this book, when accessing admin pages, we'll almost always direct you to a certain admin page by providing the url. You can always take advantage of coffee module (`Alt-d` and search term) for faster navigation.

Over the next several chapters as we build out the site and implement the theme, we'll install and enable additional modules as needed.

Development Site Settings

Next, there are a few settings we need to configure.

- Logging and errors
- Performance
- Regional settings

Logging and errors

- Navigate to `admin/config/development/logging` (or `Alt-d`, "logging")
- Set "Error messages to display" to "All messages, with backtrace information"
- Click "Save configuration"

Performance

- Navigate to `admin/config/development/performance`
- Set "Browser and proxy cache maximum age" to "<no caching>"

- Uncheck checkboxes, “Aggregate CSS files” and “Aggregate JavaScript files”
- Click “Save configuration”

Regional settings

- Navigate to `admin/config/regional/settings`
- Set “Default country”, “First day of week”, “Default time zone” appropriate for your location
- Normally the “Users may set their own time zone” checkbox is left un-checked.
- Click “Save configuration”

Summary

In this chapter we installed our Drupal site. We opted to not use the Drupal installation wizard and instead used Drush to install the site.

- We identified our site webroot and docroot directories and setup our settings files.
- Next, we downloaded and enabled some useful contrib modules.
- Lastly, we configured several site settings for development.

In the next chapter we’ll briefly discuss Drupal site configuration management and introduce a couple of useful contrib modules that enhance configuration management functionality. Drupal configuration management is essential knowledge even for the simplest sites.

Chapter 4 - Configuration Management

Note: *All code in this chapter is available in the project Git repo at Drupal 10+ Theming Chapter 4 Code . Screenshots are available at Drupal 10+ Theming Chapter 4 Screenshots.*

Drupal Configuration Management (introduced in Drupal 8) is a feature that allows developers and site administrators to manage the configuration of a Drupal site in a systematic and version-controlled way. It directly addresses managing configuration changes across different website environments, such as development, staging, and production.

Understanding and working with configuration management is essential for Drupal site building and development.

The purpose of this chapter isn't to do a deep dive into Drupal configuration management but to introduce, install and use a couple of contrib modules that greatly enhance configuration management functionality. In *Chapter 5 - Git Setup and Initial Commit* we'll see the results of using these modules.

Configuration Split

The Configuration Split module allows configuration to exported to one or more directories, enabling you to isolate configurations meant for different environments (i.e. development, staging, production).

The canonical use case is with the Devel module. The Devel module and its configuration is meant to be used in a development environment only. We don't want the module enabled in other environments (i.e. staging and production) or have its configuration exported.

Follow the below steps to install, enable and configure the Configuration Split module.

Installation

```
~/Projects/drupal-sites/arsha $ ddev composer require drupal/config_s-
plit; ddev drush en -y config_split
```

Configuration

- Navigate to `admin/config/development/configuration/config-split`
- Click on `+Add Configuration Split setting button`
- Fill out the form with the indicated settings as shown below and click “Save”. See below screenshot.

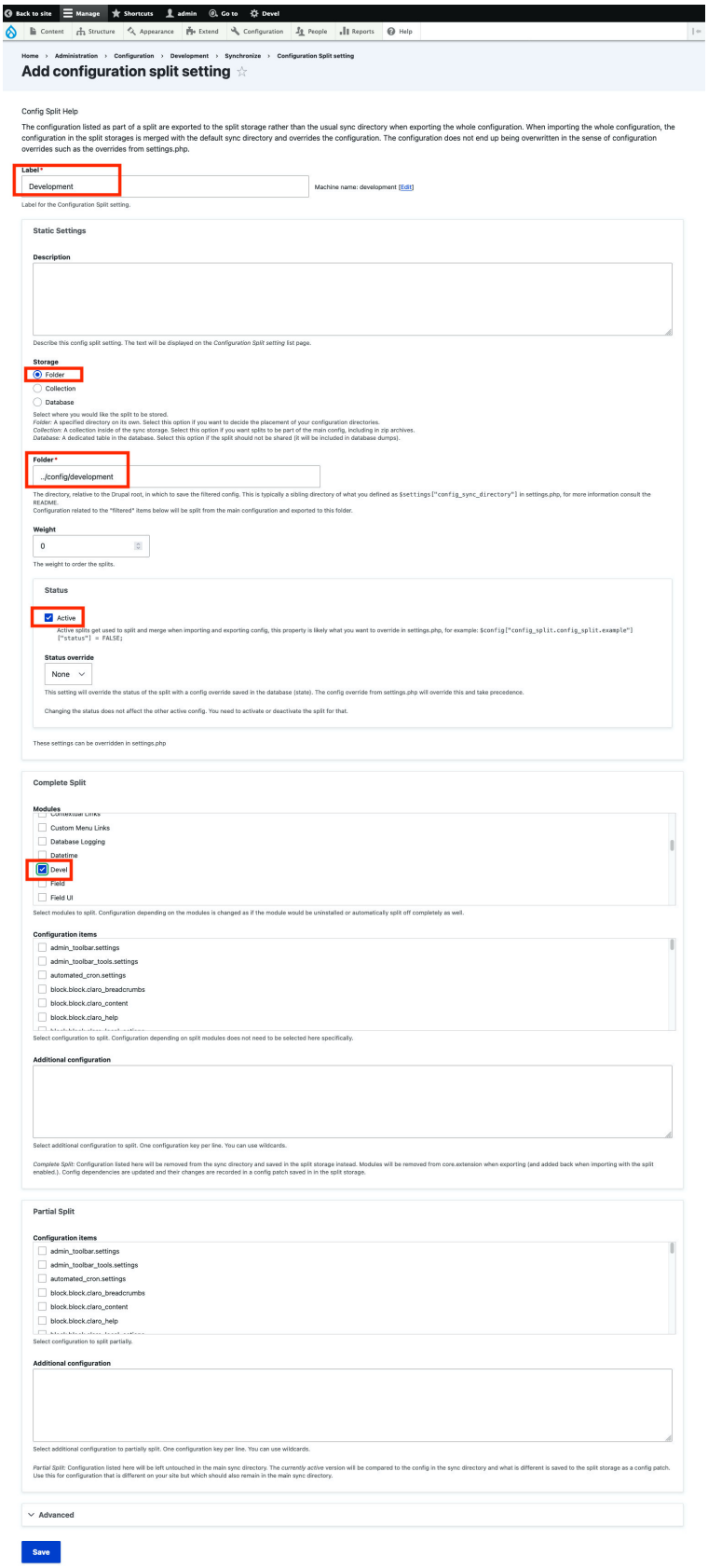

Back to site
Manage
Shortcuts
admin
Go to
Devel
Content
Structure
Appearance
Extend
Configuration
People
Reports
Help
Home › Administration › Configuration › Development › Synchronize › Configuration Split setting
Add configuration split setting
Config Split Help
The configuration listed as part of a split are exported to the split storage rather than the usual sync directory when exporting the whole configuration. When importing the whole configuration, the configuration in the split storages is merged with the default sync directory and overrides the configuration. The configuration does not end up being overwritten in the sense of configuration overrides such as the overrides from settings.php.
Label *
Development
Machine name: development [Edit]
Label for the Configuration Split setting.
Static Settings
Description
Describe this config split setting. The text will be displayed on the Configuration Split setting list page.
Storage
Folder
Collection
Database
Select where you would like the split to be stored.
Folder: A specified directory on its own. Select this option if you want to decide the placement of your configuration directories.
Collection: A collection inside of the sync storage. Select this option if you want splits to be part of the main config, including in zip archives.
Database: A dedicated table in the database. Select this option if the split should not be shared (it will be included in database dumps).
Folder *
../config/development
The directory, relative to the Drupal root, in which to save the filtered config. This is typically a sibling directory of what you defined as $settings["config_sync_directory"] in settings.php, for more information consult the README.
Configuration related to the "filtered" items below will be split from the main configuration and exported to this folder.
Weight
0
The weight to order the splits.
Status
Active
Active splits get used to split and merge when importing and exporting config, this property is likely what you want to override in settings.php, for example: $config["config_split.config_split.example"]["status"] = FALSE;
Status override
None
This setting will override the status of the split with a config override saved in the database (state). The config override from settings.php will override this and take precedence.
Changing the status does not affect the other active config. You need to activate or deactivate the split for that.
These settings can be overridden in settings.php
Complete Split
Modules
Custom Menu Links
Database Logging
Datetime
Devel
Field
Field UI
Select modules to split. Configuration depending on the modules is changed as if the module would be uninstalled or automatically split off completely as well.
Configuration items
admin_toolbar.settings
admin_toolbar_tools.settings
automated_cron.settings
block.block.claro_breadcrumbs
block.block.claro_content
block.block.claro_help
Select configuration to split. Configuration depending on split modules does not need to be selected here specifically.
Additional configuration
Select additional configuration to split. One configuration key per line. You can use wildcards.
Complete Split: Configuration listed here will be removed from the sync directory and saved in the split storage instead. Modules will be removed from core.extension when exporting (and added back when importing with the split enabled.). Config dependencies are updated and their changes are recorded in a config patch saved in in the split storage.
Partial Split
Configuration items
admin_toolbar.settings
admin_toolbar_tools.settings
automated_cron.settings
block.block.claro_breadcrumbs
block.block.claro_content
block.block.claro_help
Select configuration to split partially.
Additional configuration
Select additional configuration to partially split. One configuration key per line. You can use wildcards.
Partial Split: Configuration listed here will be left untouched in the main sync directory. The currently active version will be compared to the config in the sync directory and what is different is saved to the split storage as a config patch. Use this for configuration that is different on your site but which should also remain in the main sync directory.
Advanced
Save

Figure - ebook/screenshots/ch4/configuration-split-settings.jpg

We set up a “Development” configuration to be exported to directory `../config/development`. We choose the entire Devel module configuration to be exported to this directory. This means that the module’s configuration will not show up in other environments (i.e. staging or production).

Note: We’re using the Config Split module in a very basic way. The module can be configured to do much more. See Configuration Split Documentation for more information and use cases.

Update settings.ddev.php File

In the previous section we configured the Config Split module to export development only configuration to directory `../config/development`. In order for this work properly we need to:

- Create `config/development` and `config/sync` directories.

```
~/Projects/drupal-sites/arsha $ mkdir -p config/development config/sync
```

- Update our `setting.ddev.php` file with the correct configuration directories.

Open `web/sites/default/settings.ddev.php`. Replace the following snippet

```
// Set $settings['config_sync_directory'] if not set in settings.php.
if (empty($settings['config_sync_directory'])) {
  $settings['config_sync_directory'] = 'sites/default/files/sync';
}
```

with

```
// Set $settings['config_sync_directory'] if not set in settings.php.
if (empty($settings['config_sync_directory'])) {
  $settings['config_sync_directory'] = '../config/sync';
```

```
}

$config['config_split.config_split.development']['status'] = TRUE;
```

Before this change, production configuration would be stored in `web/sites/default/files/sync` . We've changed this directory to `../config/sync` . This will move the site configuration files out of `webroot`. Moving configuration out of `webroot` is a best practice as it ensures that site configuration won't ever be publicly accessible (when the site is made live).

`config/sync` will contain the site's production environment configuration. `config/development` will contain development environment configuration.

This is just one way to do things. It's possible for example to create a `config/staging` directory that holds configuration for a staging environment. Here though we'll keep things simple and assume we have two environments - development and production.

Config Ignore

By default, Drupal configuration management will export all site configuration to the specified directory or directories. In our case, the directories are `docroot/config/sync` and `docroot/config/development`. This configuration can then be imported into the desired environment (i.e staging site, production site) .

But what if we wanted to remove certain configuration settings from configuration management control and manually control these setting ourselves.

For example, in *Chapter 3 - Install the Site -> Development Site Settings* we enabled all error messages to be displayed on-screen (`admin/config/development/logging`) . We also disabled CSS and JavaScript file aggregation (`admin/config/development/performance`).

For a production site though, we want error messages suppressed and CSS/JavaScript file aggregation enabled. In both of these instances we'd like to manually control these settings and not let configuration management "get in the way". This is where the Config Ignore module comes in.

The Config Ignore module lets you specify configuration settings that will be ignored when configuration is imported into another environment.

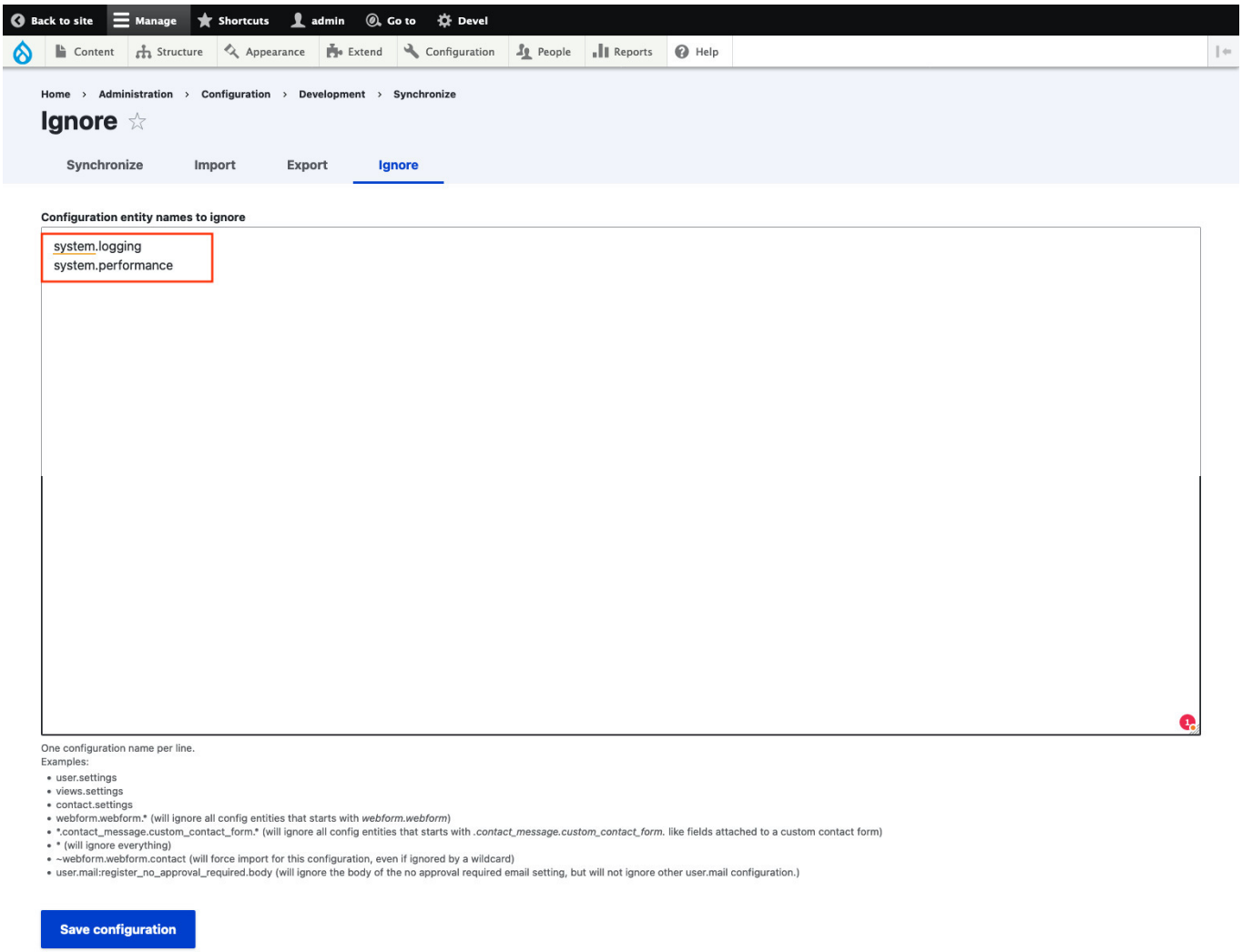
Follow the below steps to install, enable and configure the Config Ignore module.

Installation

```
~/Projects/drupal-sites/arsha $ ddev composer require drupal/config_ig-
nore; ddev drush en -y config_ignore
```

Configuration

- Navigate to `admin/config/development/configuration/ignore`
- Fill out the form as show below and click “Save configuration”.



ebook/screenshots/ch4/config-ignore-settings.jpg

We added the Logging and errors (`system.logging`) and Performance (`system.performance`) settings names (without the `.yml` extension) to the list of configurations to ignore. These settings will be ignored when site configuration is imported into another environment.

These are only two possible configuration settings we want to ignore. For a typical site you'll probably end up adding a lot more settings.

Summary

In this chapter we briefly discussed Drupal's Configuration Management system. This system allows site builders and developers to manage configuration changes systematically across different environments.

- We introduced two contrib modules, Configuration Split and Config Ignore. These modules greatly enhance configuration management functionality.
- The Configuration Split module isolates environment-specific configurations.
- The Config Ignore module excludes certain settings from being exported to different environments.

Step-by-step installation and set up for these modules was provided, along with guidance on setting up configuration directories.

With the site installation and configuration from the last two chapters, *Chapter 3 - Install the Site* and *Chapter 4 - Configuration Management* completed, we're now ready to perform our first Git commit and initial database backup.

Chapter 5 - Git Setup and Initial Commit

Note: *All code in this chapter is available in the project Git repo at [Drupal 10+ Theming Chapter 5 Code](#).*

As mentioned earlier, Git knowledge is essential for Drupal development. In this chapter we assume you're familiar with version control basics and are comfortable using Git.

.gitignore

When working with Git, one important consideration is what files should be under version control and what files should be excluded. The `.gitignore` file is used to specify files and directories that should be ***excluded*** from version control.

The Drupal installation provides the example file, `webroot/example.gitignore`. One option is to use this file as is and copy it to `docroot/.gitignore`.

Another option is to use `example.gitignore` as a starting point and create our own custom `.gitignore`. This is the option we'll choose.

Shown below is the `.gitignore` file we'll use for our project. Copy and paste it to `docroot/.gitignore`.

```
# This file contains default .gitignore rules. To use it, copy it
to .gitignore,

# and it will cause files like your settings.php and user-uploaded files
to be

# excluded from Git version control. This is a common strategy to avoid

# accidentally including private information in public repositories and
patch
```

```
# files.
#
# Because .gitignore can be specific to your site, this file has a dif-
ferent
# name; updating Drupal core will not override your custom .gitignore
file.


# Ignore core when managing all of a project's dependencies with Composer
# including Drupal core.
# core


# Ignore dependencies that are managed with Composer.
# Generally you should only ignore the root vendor directory. It's impor-
tant
# that core/assets/vendor and any other vendor directories within contrib
or
# custom module, theme, etc., are not ignored unless you purposely do so.


# We don't ignore files in the vendor directory.
# If we did we'd have to install and run composer on our production
# site to download and update these files.
# Some hosting providers may not allow us to install composer or
# other third-party packages.
# /vendor/
```

```
# Ignore site specific files and files that may contain sensitive infor-
mation.
/web/sites/*/settings*.php
/web/sites/*/services*.yml

# Ignore paths that contain user-generated content.
/web/sites/*/files
/web/sites/*/private

# Ignore multi-site test environment.
/web/sites/simpletest

# Ignore DDEV configuration files
/.ddev/

# Ignore IDE (PhpStorm and VSCode) files
/.idea/
/.vscode/



# Ignore node modules
/node_modules

# Ignore css related stuff
*.css.map
```

```
*.css.map.map


# Ignore MacOS artifacts
.DS_Store
```

Note that we put the following files under version control.

- Drupal core files under `web/`
- `vendor/` files
- Configuration files `config/`

Note: See Commit your Composer-managed files to version control for a good rationale for adding Drupal core and `vendor/` files to version control.

Now that we have a good `.gitignore` file, perform the below steps:

- Export initial configuration. Configuration will be exported to `docroot/config/sync/` and `docroot/config/development`.

```
~/Projects/drupal-sites/arsha $ ddev drush cex
```

Note the contents of `docroot/config/development` contains the Devel module configuration as we "split off" this configuration . *See Chapter 4 - Configuration Management -> Configuration Split*.

```
~/Projects/drupal-sites/arsha $ ls config/development/
devel.settings.ymldevel.toolbar.settings.ymlsystem.menu.devel.yml
```

- Perform the first commit.

```
~/Projects/drupal-sites/arsha $ git init
~/Projects/drupal-sites/arsha $ git config user.name <your_user_name>
```

```
~/Projects/drupal-sites/arsha $ git config user.email <your_email_address>

~/Projects/drupal-sites/arsha $ git add .

~/Projects/drupal-sites/arsha $ git ci -m "Initial commit"


~/Projects/drupal-sites/arsha $ git st

On branch main

nothing to commit, working tree clean
```

- Export initial database snapshot

```
~/Projects/drupal-sites/arsha $ ddev export-db -d db -f ~/Desktop/Website-Backups/arsha/arsha-db-ch5-$(date +"%m-%d-%Y-%H-%M-%S").sql.gz
```

For easier backups, the `run-site-backup.sh` script can be used.

```
~/Projects/drupal-sites/arsha $ cat run-site-backup.sh

#!/usr/bin/env bash

# Backup site

DDEV=/usr/local/bin/ddev


# Set to desired location for site backups

backup_dir=$HOME/Desktop/Website-Backups

site_dir=${PWD}

site=$(basename ${site_dir})


echo "Site is ${site}"
```

```
${DDEV} export-db -d db -f ${backup_dir}/${site}/${site}-db-$(date +"%m-
%d-%Y-%H-%M-%S").sql.gz
```

- Run backup using `run-site-backup.sh` script.

```
~/Projects/drupal-sites/arsha $ ./run-site-backup.sh

Site is arsha

Wrote database dump from project 'arsha' database 'db' to file /Users/
<user_name>/Desktop/Website-Backups/arsha/arsha-
db-08-03-2023-17-01-56.sql.gz in gzip format.
```

In the above examples, we store our site database snapshots in directory `~/Desktop/Website-Backups/arsha/` but of course you can save the backups anywhere you like. ***It's prudent to take frequent site backups***.

Lastly we'll take an initial "snapshot" of our site DDEV environment. In case our DDEV environment gets corrupted or stops working we can use `ddev snapshot restore <snapshot_name>` to restore the environment.

```
~/Projects/drupal-sites/arsha $ ddev snapshot --name 08_03_23
```

Summary

In this chapter we performed our initial Git commit.

- We began by creating an appropriate `.gitignore` file. This file excludes sensitive or unnecessary files, such as local settings files, IDE files, and development artifacts from version control.

- Next, we exported configuration files for both production and development environments using DDEV and Drush. These files will be under version control.

- Once all files are in place we performed the first commit.

- We then performed an initial database backup of our site. ***It's important to perform regular database backups***. This can be done manually using DDEV or using the provided `run-site-backup.sh` script.

- Lastly, the chapter introduced DDEV snapshots. Snapshots should be periodically performed to safeguard against DDEV environment corruption or failures.

In the next chapter we'll continue our Drupal theming journey by reviewing theming basics, theme selection, theme installation and theming environment setup.

Chapter 6 - Drupal Theming Overview

Note: *All code in this chapter is available in the project Git repo at Drupal 10+ Theming Chapter 6 Code . Screenshots are available at Drupal 10+ Theming Chapter 6 Screenshots.*

A Drupal theme, simply put, determines the ***look and feel*** of a Drupal site. Drupal provides many Drupal 10+ compatible contributed themes. These themes can be used as is or as a base theme to be customized. An example contributed theme is Bootstrap5.

When choosing a theme for a Drupal site, many options are available.

1. Use a contributed theme available from drupal.org (contributed themes).

However, the number of themes available is limited compared to themes available on other platforms like Wordpress or the many website builders (Squarespace, Wix etc.). Additionally a lot of contributed themes are just base / starter themes that require a lot of customization (added CSS and / or JavaScript) to produce a finished product.

2. Use a theme from a third party vendor. The following vendors offer Drupal themes. A web search would probably find more.
    - Drupar
    - YG Themes
    - DrupalThemes.io
    - MORE THAN (just) THEMES

These vendors typically offer free and paid (premium) versions of their themes. The premium versions include the entire package (including a database) needed to build a demo site using the theme and extended / premium support.

3. Use a generic website template and convert it to a Drupal theme. Many vendors offer website templates - free and/or paid. Below is a partial list of vendors, many others exist.

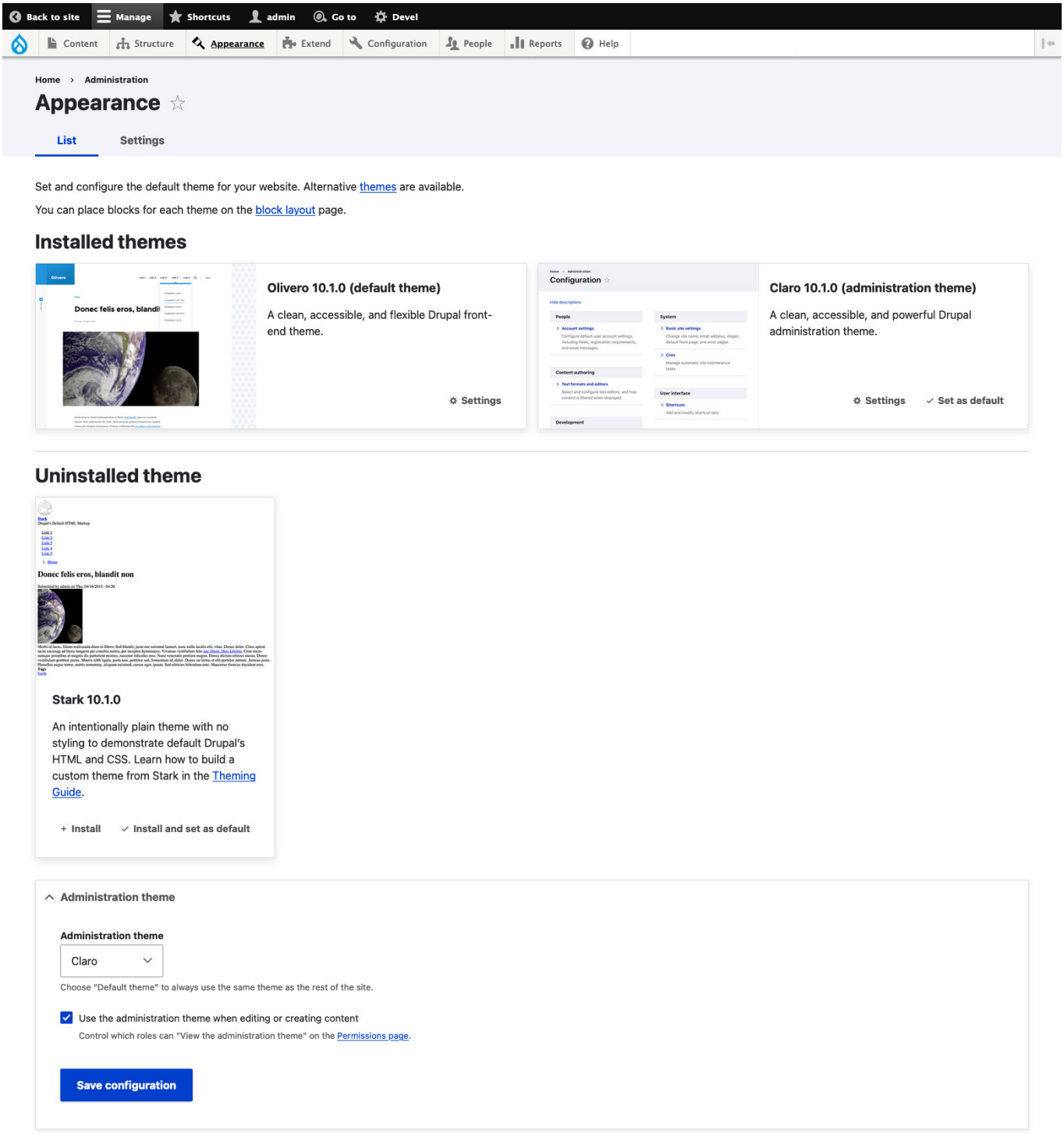
- ThemeForest
- ThemeFisher
- BootstrapMade
- HTML5 UP

4. The last option, mostly applicable for medium to large organizations, is to produce an in-house custom website design and build a custom Drupal theme from that design.

For our site we're using option 3. As mentioned in *Chapter 1 - The Website We'll Be Building,* we'll be using the Arsha template from BootstrapMade and converting it to a Drupal theme,

Theme Administration

Themes are administered from the site Appearance page. Navigate to `admin/appearance`. See below screenshot.



ebook/screenshots/ch6/appearance-page.jpg

Out of the box (using the standard installation profile) a Drupal 10 site comes with two installed themes:

- Olivero (default / frontend theme)
- Claro (administrative / backend theme)

A third theme, Stark, is provided but by default is uninstalled.

Olivero is the “frontend” theme and is the theme used in the non-administrative pages of the site. Non logged in users will only see the Olivero theme. Claro is the “backend” theme and is used for all administrative pages.

This dual theme structure allows a clean separation between the frontend and backend parts of Drupal. This allows navigation through the Drupal admin pages with a known stable theme and avoids any page navigation issues due to frontend theme bugs and errors.

The Stark theme (uninstalled by default) is a “bare bones” theme meant to be used a base theme for custom theme development.

Theme Settings

Note the “Settings” tab on the Appearance page (`admin/appearance`). At this point in our development we can keep the default settings. In latter chapters, we’ll update these settings for our theme.

Theme Components

Implementing a theme involves several components

1. `<theme-name>.info.yml` file
2. `<theme-name>.libraries.yml` file

3. CSS, Javascript and media (image, video, audio) and font files
4. `<theme-name>.theme` file (optional)
5. `<theme-name>.theme-settings.php` file (optional)
6. TWIG Template files (`page.html.twig` and many others)
7. Regions and Blocks

Theming Drupal on Drupal.org (though in need of updating) is the canonical source for Drupal theming information and is required reading for theme development. It discusses all the theme components listed above.

A good supplementary reference is the article Anatomy of a Drupal 8 Theme.

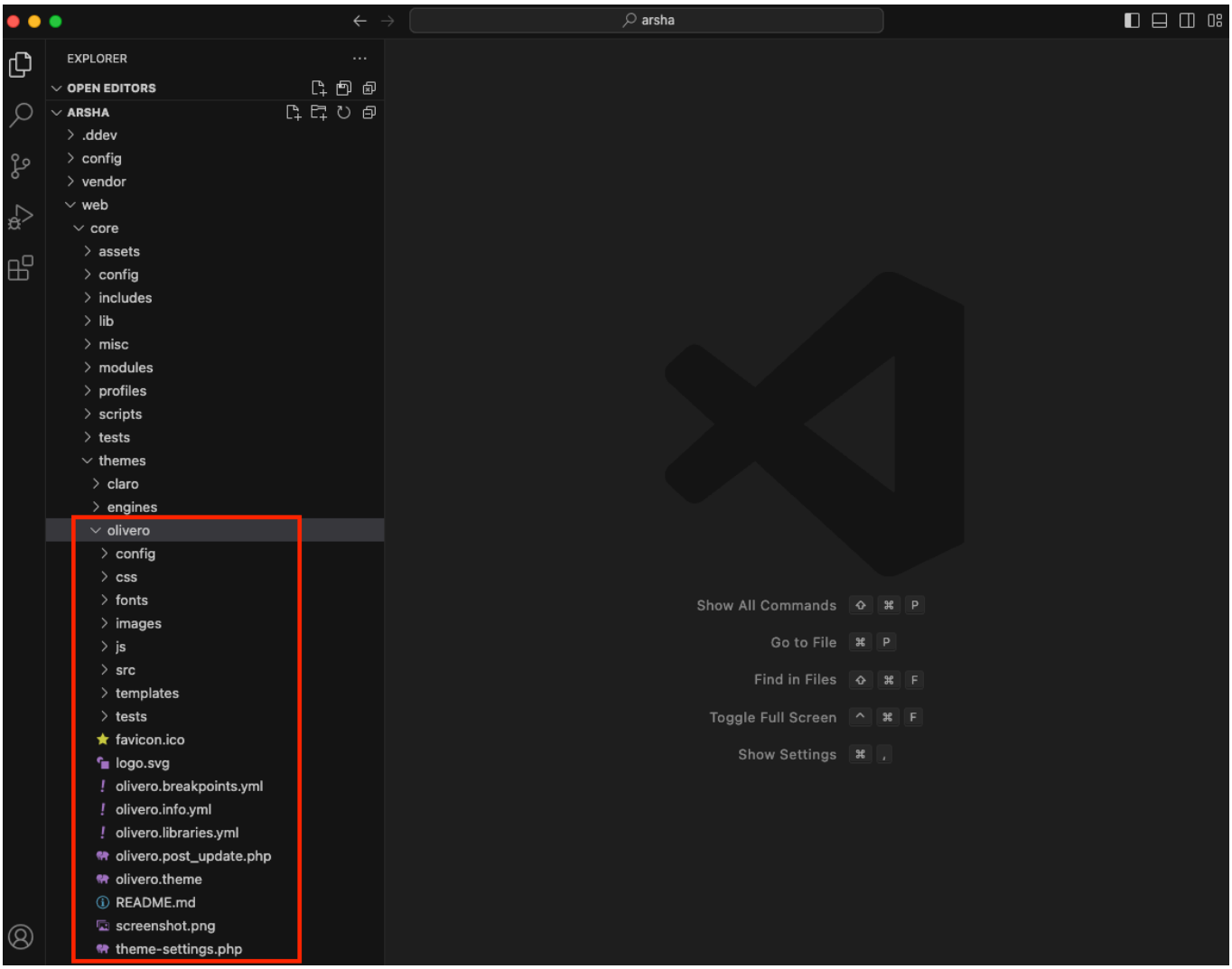
Theme files are located in one of two places.

- `web/core/themes`
- `web/themes`

Drupal core themes are in `web/core/themes`. Here you'll find the files for the Olivero and Claro themes mentioned above.

Contributed themes (that you install) and custom themes (that you create) are located in `web/themes/contrib` and `web/themes/custom` respectively.

Below is a screenshot of the Olivero theme directory structure and files.



ebook/screenshots/ch6/olivero-theme-files.png

You can browse through these files to get an idea of their contents and functionality while reading Theming Drupal. In later chapters we'll get our hands dirty and create the theme files for our Astra theme.

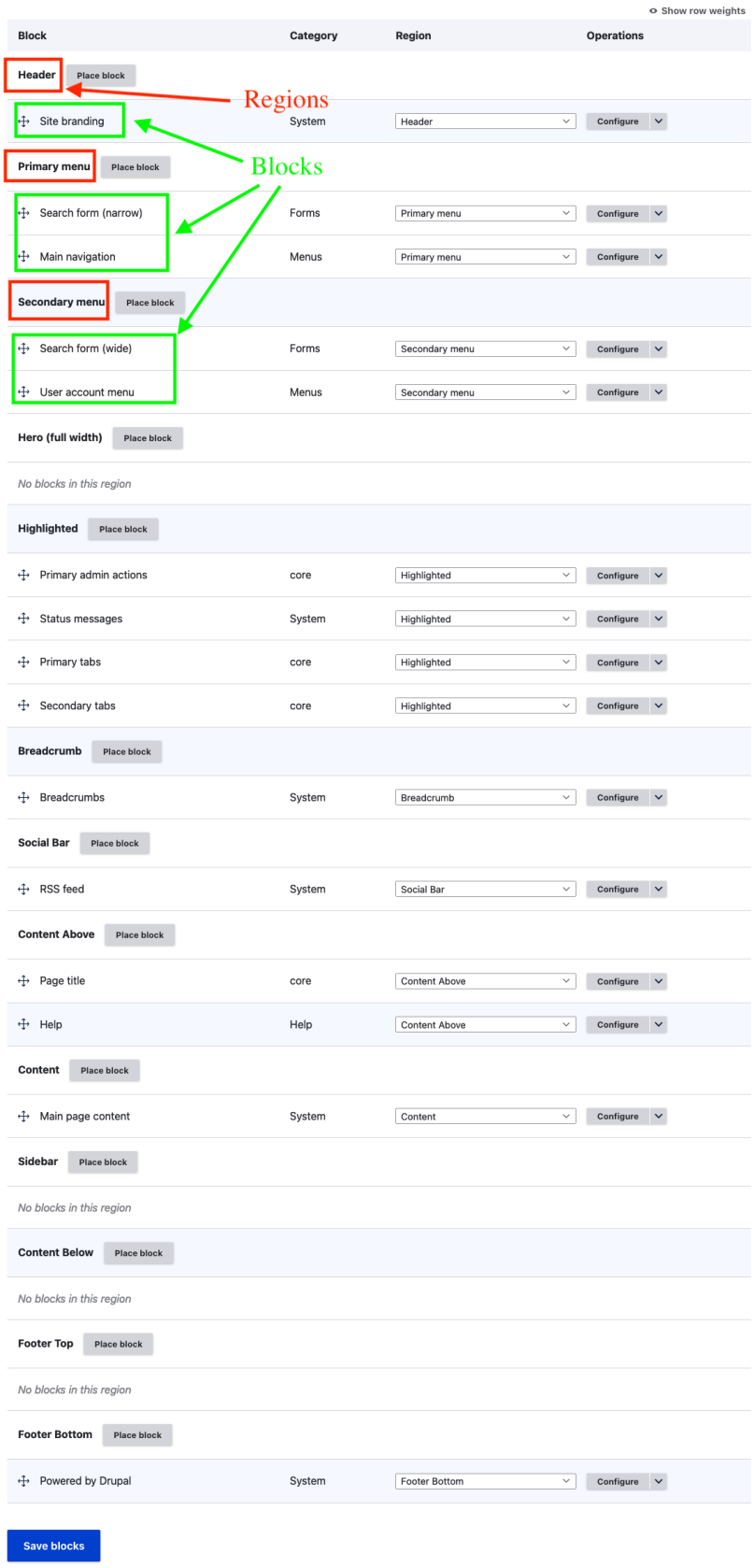
Regions and Blocks

Regions are areas of a page into which content can be placed. Regions are defined in a theme's `*.info.yml` file, `olivero.info.yml` for the Olivero theme. Below is an excerpt from the `olivero.info.yml` file showing how regions are defined.

```
regions:
  header: Header
  primary_menu: Primary menu
  secondary_menu: Secondary menu
  hero: Hero (full width)
  highlighted: Highlighted
  breadcrumb: Breadcrumb
  social: Social Bar
  content_above: Content Above
  content: Content
  sidebar: Sidebar
  content_below: Content Below
  footer_top: Footer Top
  footer_bottom: Footer Bottom
```

Olivero defines several regions, “Header”, “Primary menu”, “Second menu”, “Hero (full width)” etc.

Website content is assigned to regions via *Blocks*. Blocks contain individual pieces of a site’s content. Navigate to `admin/structure/block` to view the Olivero regions and blocks assigned to the regions. See below screenshot.


Show row weights
Block
Category
Region
Operations
Header
Place block
Regions
Site branding
System
Header
Configure
Primary menu
Place block
Blocks
Search form (narrow)
Forms
Primary menu
Configure
Main navigation
Menus
Primary menu
Configure
Secondary menu
Place block
Search form (wide)
Forms
Secondary menu
Configure
User account menu
Menus
Secondary menu
Configure
Hero (full width)
Place block
No blocks in this region
Highlighted
Place block
Primary admin actions
core
Highlighted
Configure
Status messages
System
Highlighted
Configure
Primary tabs
core
Highlighted
Configure
Secondary tabs
core
Highlighted
Configure
Breadcrumb
Place block
Breadcrumbs
System
Breadcrumb
Configure
Social Bar
Place block
RSS feed
System
Social Bar
Configure
Content Above
Place block
Page title
core
Content Above
Configure
Help
Help
Content Above
Configure
Content
Place block
Main page content
System
Content
Configure
Sidebar
Place block
No blocks in this region
Content Below
Place block
No blocks in this region
Footer Top
Place block
No blocks in this region
Footer Bottom
Place block
Powered by Drupal
System
Footer Bottom
Configure
Save blocks

ebook/screenshots/ch6/olivero-regions-blocks.png

In the screenshot, the first few regions and blocks are marked. All blocks shown are provided by Drupal out of the box and have been assigned to the regions by default during site installation.

Theme Development Setup

During theme development there's a few things to do that make development easier.

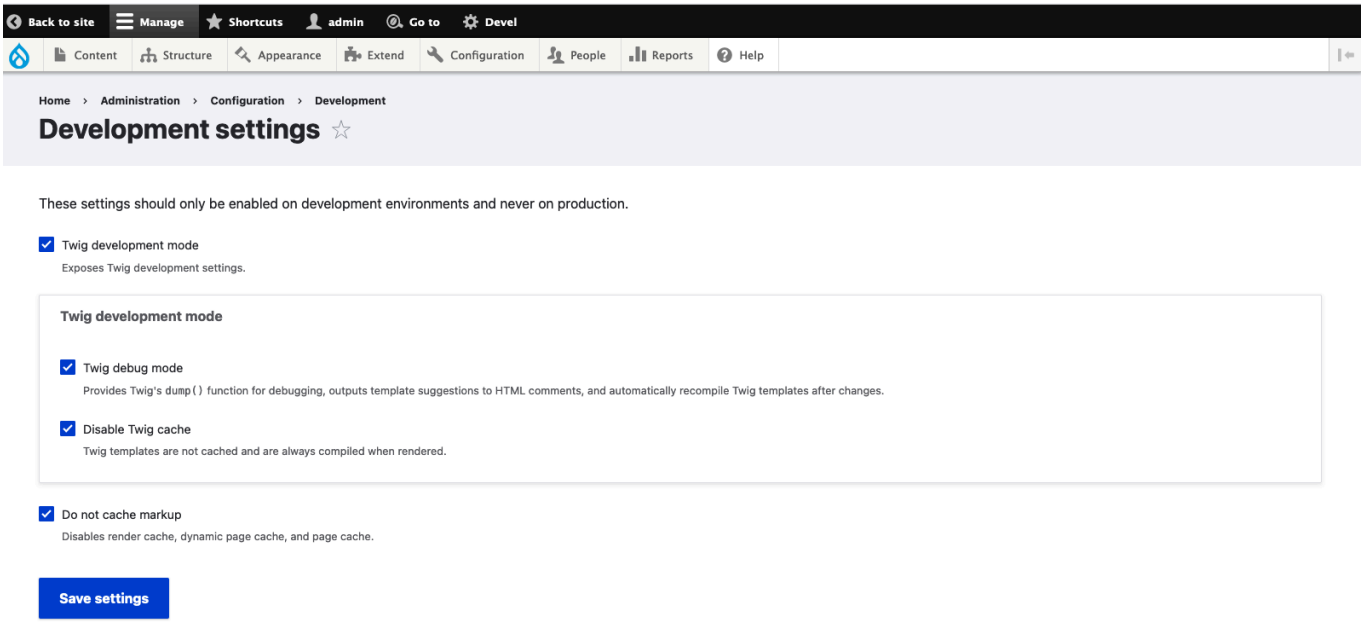
- Turn on Twig template debugging
- Disable Twig cache
- Disable Drupal render, dynamic page and page caches
- Disable CSS and JavaScript file aggregation

Turning on Twig template debugging allows you to see which template file is being used to render a page, block, form, etc. Disabling the above mentioned caches reduces the need to clear the caches after making file changes.

Prior to Drupal 10.1, these changes were done by modifying the files `web/sites/default/services.yml` and `web/sites/default/settings.local.php`. See Debugging Twig Templates and Disable caching for more information.

New in Drupal 10.1 is a new admin page that controls these settings.

- Navigate to `admin/config/development/settings` or with the coffee module, `Alt-d` and "Development settings".
- Check the checkboxes as shown in the below screenshot



ebook/screenshots/ch6/theme-development-settings.png

- Click “Save settings”
- Clear caches (`Alt-d` , “Flush all caches” or `drush cr`)

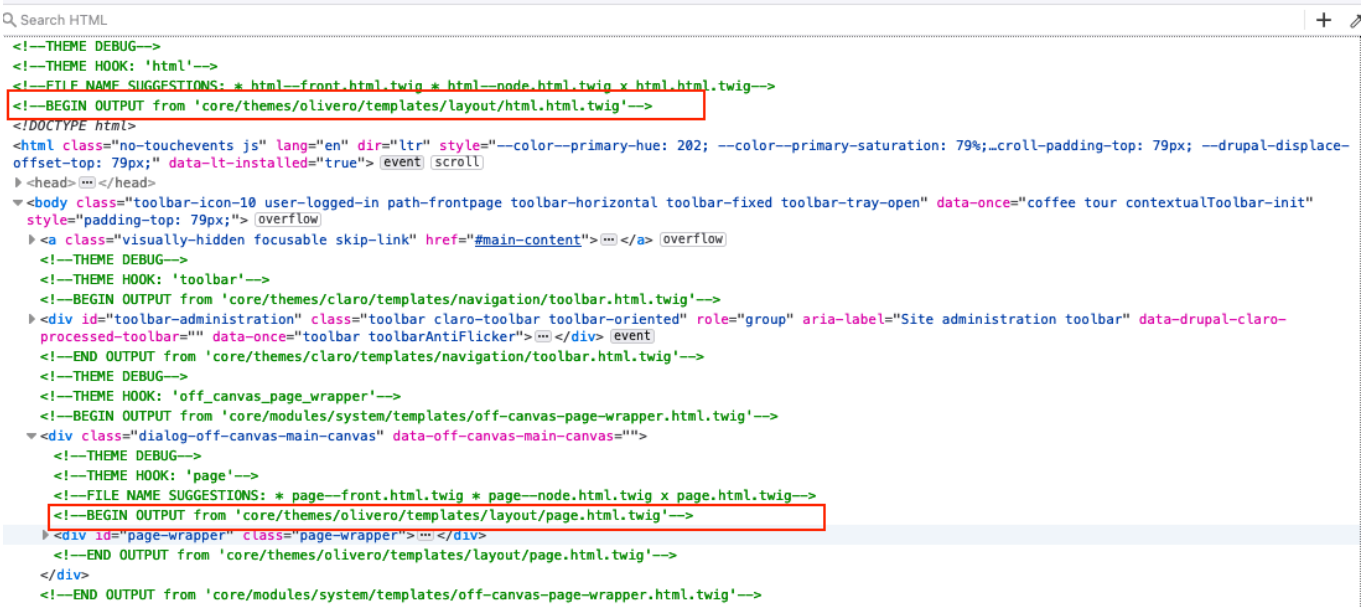
Browser Inspector

Throughout our site build and theme development, we’ll frequently use the web browser Inspector to view page markup. All major browsers, Google Chrome, Firefox, Microsoft Edge, include this feature as part of their Web Developer Tools.

Being familiar with using the Inspector is vital for Drupal theme development and overall website development. Refer to the many web articles or YouTube tutorials for information on using the Inspector and other browser Web Developer Tools.

Navigate to the front page of the site and invoke your web browser Inspector (**mouse right click and then “Inspect”**) to view the page markup.

As the below screenshot shows, with Twig debugging enabled, the Twig template files used for page markup show up as comments.



ebook/screenshots/ch6/front-page-template-files-olivero.png

Note that the following Olivero theme files are being used for the overall top level page markup

- `core/themes/olivero/templates/layout/html.html.twig`
- `core/themes/olivero/templates/layout/page.html.twig`

Disable CSS and JavaScript File Aggregation

As a final step, we'll disable CSS and JavaScript file aggregation. This will let us see the individual CSS and JavaScript files used on a page. Otherwise the files will be aggregated together and compressed.

In `web/sites/default/settings.local.php`, ensure that file aggregation is off as shown below.

```
/*
```

```
 * Disable CSS and JS aggregation.
 */
$config['system.performance']['css']['preprocess'] = FALSE;
$config['system.performance']['js']['preprocess'] = FALSE;
```

If you need to change the settings, make sure you clear caches afterwards.

Summary

This chapter provided an overview of Drupal 10 theming, from selecting and managing themes, to discussing theme components.

- We began by discussing different options for themes, including contributed themes from Drupal.org, premium themes from third-party vendors, and custom-built themes.
- Next, we briefly discussed the dual-theme structure of Drupal. Out of the box, a Drupal installation uses the Olivero theme for the frontend and the Claro theme for the back-end. This structure ensures a stable separation between site visitor pages and administrative pages.
- The key theme components are then discussed, including the theme `.info.yml` and `.libraries.yml` files, Twig template files, theme regions and content blocks.
- Lastly, we set up our site for easier theme development, including enabling Twig debugging and disabling caches.

In the next chapter we'll do more prep work and continue our quest to build a Drupal theme from the Arsha template.

Appendix C - Contrib Modules

Below is a chapter by chapter list of contrib modules used in the site we build.

Chapter 3 - Install the Site

- Admin Toolbar
- Coffee
- Pathauto
- Smart Trim
- Allowed Formats

  Removed in *Chapter 11 - Building Out the Page Structure -> Add Dummy Content* and *Appendix A - Drupal Component Configuration -> Text Formats*

- Devel

Chapter 4 - Configuration Management

- Configuration Split
- Config Ignore

Chapter 12 - Front Page Theming (Part 1)

- Webform
- Social Media Links Block and Field

Chapter 13 - Front Page Theming (Part 2)

- Paragraphs
- Fences - Semantic field markup and classes
- Twig Tweak

Chapter 14 - Front Page Theming (Part 3)

- Current Page Crumb

Chapter 16 - Front Page Theming (Part 4)

- Formdazzle!

Appendix A - Drupal Component Configuration

- Simple Google Maps
- Weight
- Views Reference Field
- cweagans/composer-patches
- Block field

Appendix D - Resources

This chapter lists some of the resources that were used in writing this eBook.

Bootstrap

- How to… Bootstrap
- Bootstrap 5 documentation

Twig

- Twig

Drupal.org Resources

- Theme System Overview
- Theming Drupal
- Twig in Drupal
- Twig Template naming conventions
- CSS File Organization
- Preprocessing and modifying attributes in a .theme file
- Render API overview

Drupal Development

- Drupal at your fingertips
- Drupal Answers

Miscellaneous Resources

- Anatomy of a Drupal 8 Theme
- Drupal Theming Do's and Don'ts
- Managing CSS and JavaScript files in Drupal 8 with Libraries
- How to use Hooks for building Drupal 8 theme
- An Introduction to Drupal's Render Arrays
- Standard Drupal Render Array Properties
- Video | Aha! Understanding and Using Render Arrays in Drupal 8
- New to Drupal: Improved Dumping of Twig Variables!