



Summer Fellowship Report
On
DRUPAL (Web Development)

Submitted by
Ajeet Kumar Yadav

Under the guidance of

Prof. Pradeep Varma
Computer Science Engineering Department
IIT Bombay

21st May -12th July 2019

ACKNOWLEDGMENT

This report has been prepared for the internship that has been done in the National Virtual Library of India (NVLI), Janpath Office (National Mission on Libraries Janpath Bhawan), Delhi in order to study the practical aspects of the Drupal its implementation in the real field with the purpose of learning and exploring in the mentioned field and for professional development.

The aim of this internship is to be familiar with the implication of Drupal and uses of concerned theoretical knowledge and clarifying the career goals so, I have successfully completed the internship and compiled this report as the summary.

I would like to express my sincere gratitude to my project head Prof. Pradeep Varma who has given me this valuable opportunity to work in such a learning environment to get industrial-based experience. I am also grateful to the enthusiastic team of NVLI project especially the Technical team. The Technical team comprising of three great mentors Mr Pratish Jha, Mr Manish Kala, Miss. Sonpriya Rawat their constant supervision and support was truly helpful in completing my tasks and my internship.

Ajeet Kumar Yadav

Place: New Delhi

Date: 12th July 2019



CONTENTS

1. ACKNOWLEDGMENT

2. CONTENTS

3. INTRODUCTION

- i. The two code sets used by every Drupal site: Codebase, and Database
- ii. Flexibility, meet Simplicity
- iii. The Drupal flow

4. Dependency manager for PHP

- i. The role of Composer
- ii. Installing Composer locally
- iii. Speeding up tasks using Drush
- iv. Installing Drush globally

5. Task 1 : Exploring Views and Storing views count on nodes

- i. Concept: Uses of Views
- ii. Looking at the Views administration page
- iii. Views settings
- iv. Views basic settings

6. Task 2 : Implementing Like/Dislike on node

- i. Like/Dislike Module



7. Task 3 : Implementing Comment count on node

- i. For comment count

8. Task 4 : Implementing Share Button for Social Media Platform

- i. AddToAny Share Buttons Module

9. Task 5 : Implementing Library name from which the library data is coming

- i. Twig Field Value

10. Task 6 : To Explore the Taxonomy use on an instance

- i. Steps for creating taxonomy

11. Task 7 : To explore the theming of bootstrap And To create custom login theming

- i. Theming
- ii. Twig
- iii. Sub-Theming
- iv. Twig Template naming conventions

12. REFERENCES



INTRODUCTION

Drupal is a free, open-source content management system (CMS) with a large, supportive community. This guide is for people who are just sorting out what Drupal is and what you can do with it. It focuses on resources that provide an informative overview instead of hands-on tutorials. It is designed to give some context and history of the Drupal project, along with the major concepts involved in building a site with it. If you want to begin learning Drupal hands-on, the best start is the Drupal 8 User Guide, which walks you through the foundations of working with Drupal.

The two code sets used by every Drupal site: Codebase, and Database

Codebase: These are the files and folders you get when you download Drupal. These files in the codebase are responsible for creating and managing all of your site's content, such as articles, or user comments, The content itself (as well as settings and config) is stored in the database.

When you want to duplicate, or move a Drupal site, you need to grab a copy of both the codebase, and database.

Flexibility, meet Simplicity

Solutions for content management struggle to balance flexibility and simplicity. If a solution is simple, it can only be used for a single purpose and if it is flexible, it may be too difficult for newcomers to learn.

The average content management system (CMS) is like a toy truck—specific assumptions have been made about how it will be used, and these assumptions are difficult to override. Content management frameworks, on the other hand, are like the raw materials needed to make any toy—no assumptions have been made about



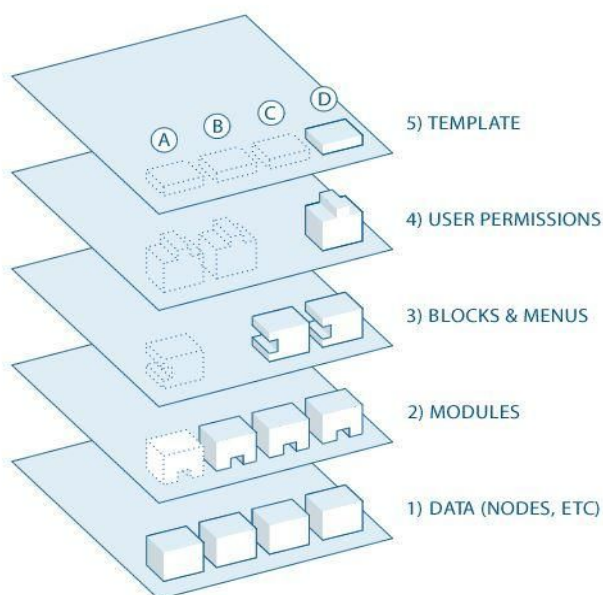
how they'll be used and the builder needs expert technical knowledge in order to make anything at all.

Drupal is designed to be the perfect content management solution provider for non-technical users who need both simplicity and flexibility. It accomplishes this through its modular approach of site building. Unlike other CMSs, Drupal isn't a prefabricated toy truck, but rather a collection of wheels, windshields, axles, frames, etc. that a toy-maker can easily connect together. With Drupal, a maker could create a toy truck and similarly can also create a toy such as airplane, submarine, or robot. For this reason, Drupal may be described as both a content management system and a content management framework—one unified system that strives to have the strengths of both, without their deficiencies.

So, whether a site builder is looking to create a news site, online store, social network, blog, wiki, or anything else, it's just a matter of combining the right modules. The only limitations are the creator's imagination.

The Drupal flow

If you want to go deeper with Drupal, you should understand how information flows between the system's layers. There are five main layers to consider:



Dependency manager for PHP

The role of Composer

Drupal 8 and each minor version introduces new features and functionality: everything from moving the most commonly used third-party modules into its core to the introduction of an object-oriented PHP framework. These improvements also introduced the **Symfony framework** that brings in the ability to use a dependency management tool called Composer.

Composer (<https://getcomposer.org/>) is a dependency manager for PHP that allows us to perform a multitude of tasks: everything from creating a Drupal project to declaring libraries and even installing contributed modules, just to name a few. The advantage of using Composer is that it allows us to quickly install and update dependencies by simply running a few commands. These configurations are then stored within a `composer.json` file that can be shared with other developers to quickly set up identical Drupal instances.

If you are new to Composer, let's take a moment to discuss how to go about installing Composer for the first time within a local environment.

Installing Composer locally

Composer can be installed on Windows, Linux, Unix, and OS X. For this example, we will be following the install found at <https://getcomposer.org/download/>. Ensure that you take a look at the Getting Started documentation that corresponds with your operating system.

Begin by opening a new Terminal window. By default, our Terminal window should place us in the user directory. We can then continue by executing the following four commands:

1. Download Composer installer to the local directory:

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
```

2. Verify the installer:

Since Composer versions are often updated, it is important to refer to the date on the



Download Composer page to ensure that the preceding hash file is the most current one.

3. Run the installer:

```
php composer-setup.php
```

4. Remove the installer:

```
php -r "unlink('composer-setup.php');"
```

5. Composer is now installed locally, and we can verify this by executing the following command within a Terminal window:

```
php composer.phar
```

The challenge with having Composer installed locally is that it restricts us from using it outside the current user directory. In most cases, we will be creating projects outside of our user directory, so having the ability to globally use Composer quickly becomes a necessity.

Speeding up tasks using Drush

Drush (<http://www.drush.org/en/master/>) is a command-line shell and Unix-scripting interface that allows us to interact with Drupal. Drush gives us the ability to use the command line to accomplish tasks quickly, without the need to rely on the Drupal admin UI. As part of the composer install, our project has the latest version of Drush installed automatically.

Executing a Drush command is typically as easy as typing the word drush within a Terminal window.

However, the challenge of having a per-project instance of Drush is in the way we are forced to currently execute Drush commands. Since the drush executable is located within the projects

/vendor/bin/drush folder, if we are within the root of our project, we execute drush by entering the following within the Terminal window:

```
./vendor/bin/drush
```

The problem is the path can easily change; if, for instance, we are in the /web root, the same command would be:

```
../vendor/bin/drush
```



Notice the two dots indicating one must traverse up a level to locate the /vendor folder.

This is not ideal when we will be using Drush quite frequently to perform various tasks. We can resolve this in a couple of different ways.

Installing Drush globally

Installing Drush globally varies based on the operating system or AMP stack, as there is a

dependency on PHP 5.5.9 or higher. This dependency will be satisfied in most cases, but ensure that you verify the version of PHP that is available.

Begin by opening the Terminal window, changing into the user directory, and executing the following commands:

1. Verify that Composer is installed:

composer

2. Add Composer's bin directory to the system path:

export PATH="\$HOME/.composer/vendor/bin:\$PATH"

3. Install the latest stable release:

composer global require drush/drush

4. Verify that Drush works:

drush status

5. Now that Drush has been installed globally, we can easily ensure that we always have the latest version by running this:

composer global update

The list of Drush commands is quite long, but it does provide us with the ability to perform almost any action we may need when working on a Drupal project. Some simple commands that we will commonly use throughout the book are clearing cache, managing configurations, and even installing Drupal. For a list of all the various commands, we can browse Drush Commands at <https://drushcommands.com/>.



Task 1 : Exploring Views and Storing views count on nodes

Concept: Uses of Views

A view is a listing of content on a website. The core Views module handles the display of views, and the core Views UI module allows you to create and edit them in the administrative interface. When you define views, you are interested in taking data from your website and displaying it to the user.

A listing created by a view can be in any of the following forms:

1. Table with sortable fields
2. Grid layouts
3. Teasers or pictures that link to articles
4. Blocks
5. JSON output
6. RSS feeds
7. Calendars
8. On-screen slideshows

Step 1: Go to **Manage >> Structure >> Views >> click on +Add new view**



Structure ☆

Home » Administration

- ▶ **Block layout**
Configure what block content appears in your site's sidebars and other regions.
- ▶ **Comment types**
Manage form and display settings of comments.
- ▶ **Contact forms**
Create and manage contact forms.
- ▶ **Content types**
Create content types and manage their default settings.
- ▶ **Display modes**
Configure what displays are available for your content and forms.
- ▶ **Menus**
Manage menus and menu links.
- ▶ **Taxonomy**
Manage tagging, categorization, and classification of your content.
- ▶ **Views**
Manage customized lists of content.

Step 2: Clicking on **+Add new view** redirects to “**Add new view**” page. The page has **View Basic information** fieldset that contain **View name** and **Description**.

The screenshot shows the Drupal administration interface for creating a new view. The top navigation bar includes 'Back to site', 'Manage', 'Shortcuts', and 'admin'. Below this is a secondary navigation bar with icons for 'Content', 'Structure', 'Appearance', 'Extend', 'Configuration', 'People', 'Reports', and 'Help'. The main content area is titled 'VIEW BASIC INFORMATION' and contains the following fields:

- View name ***: A text input field containing 'latest News'. To its right, it says 'Machine name: latest_news [Edit]'.
- Description**: A checked checkbox followed by a text input field containing 'listing all latest news here'.

Below the 'VIEW BASIC INFORMATION' fieldset are three other fieldsets:

- VIEW SETTINGS**: Contains 'Show:' with a dropdown menu set to 'Content', 'of type:' with a dropdown menu set to 'All', and 'sorted by:' with a dropdown menu set to 'Newest first'.
- PAGE SETTINGS**: Contains a checkbox for 'Create a page' which is currently unchecked.
- BLOCK SETTINGS**: Contains a checkbox for 'Create a block' which is currently unchecked.

At the bottom of the form are two buttons: 'Save and edit' (in blue) and 'Cancel' (in grey).



Section 2 has **VIEW SETTING** which has option to show, type, and sorted by. Where, **Show** means what would you like to Display, Comment, Log entries, Files, Content, Content revision, Taxonomy Terms, Users, Link Count, Custom block revision or Custom block.

Of type holds bundle type either Article, Basic page, custom content type or All.

Step 3: Once you have saved the view page data, page will be redirected to view Display configuration section having **page** and **block UI**.

Just like Drupal 7 views contrib module, Drupal 8 views also has similar UI.

As we can see here view has multiple configuration.

Title : set the title for view page.

Format: How data has to be displayed on the site.

Fields: what are fields you would like to display.

Filter criteria: allow filter by type as well as option to expose the same.

Sort criteria: sorting of field date ascending or descending manner.

page Settings: set the page path, assign to menu if required, view permission.

Header: custom header for the view page.

Footer: custom footer for the view page.

No Result Behaviour: In case if view query fetches no data or empty set, we can set option to override and display custom message to user e.g. no search result available, please try again.

Pager: allow site to create pagination on view page.

Advanced: As the label suggest, this section has advanced options like CONTEXTUAL FILTERS , RELATIONSHIPS, EXPOSED FORM, OTHER etc.



Looking at the Views administration page

Views ☆

List Settings

Home » Administration » Structure

[+ Add view](#)

Filter by view name, machine name, description, or display path

Enabled

VIEW NAME	MACHINE NAME	DESCRIPTION	DISPLAYS	OPERATIONS
Comments	comment	Find and manage comments.	Page (/admin/content/comment) Page (/admin/content/comment/approval)	Edit ▾
Content	content	Find and manage content.	Page (/admin/content)	Edit ▾
Custom block library	block_content	Find and manage custom blocks.	Page (/admin/structure/block/block-content)	Edit ▾
Files	files	Find and manage files.	Page (/admin/content/files/usage/%) Page (/admin/content/files)	Edit ▾
Frontpage	frontpage	All content promoted to the front page.	Feed (/rss.xml) Page (/node)	Edit ▾
Media	media		Page (/admin/content/media)	Edit ▾
military forces	military_forces		Page (/military-forces)	Edit ▾
People	user_admin_people	Find and manage people interacting with your site.	Page (/admin/people)	Edit ▾
Recent comments	comments_recent	Recent comments.	Block	Edit ▾
Recent content	content_recent	Recent content.	Block	Edit ▾

Views settings

Advanced Views settings ☆

List Settings

Basic **Advanced**

Home » Administration » Structure » Views » Views settings

▼ CACHING

Disable views data caching
Views caches data about tables, modules and views available, to increase performance. By checking this box, Views will skip this cache and always rebuild this data when needed. This can have a serious performance impact on your site.

[Clear Views' cache](#)

▼ DEBUGGING

Add Views signature to all SQL queries
All Views-generated queries will include the name of the views and display 'view-name:display-name' as a string at the end of the SELECT clause. This makes identifying Views queries in database server logs simpler, but should only be used when troubleshooting.

[Save configuration](#)



Views basic settings

Views settings ☆

List Settings

Basic Advanced

Home » Administration » Structure » Views

- Always show the master (default) display
- Always show advanced display settings
- Allow embedded displays
Embedded displays can be used in code via `views_embed_view()`.

Label for "Any" value on non-required single-select exposed filters

<Any>

▼ LIVE PREVIEW SETTINGS

- Automatically update preview on changes
- Show information and statistics about the view during live preview
- Show the SQL query
- Show performance statistics
- Show other queries run during render during live preview
Drupal has the potential to run many queries while a view is being rendered. Checking this box will display every query run during view render as part of the live preview.

Save configuration

For view count

Module needed: Statistics

1. The module is already installed in drupal by default.
2. Go to extend and enable the module Statistics.
3. Go to Configuration->Statistics then tick the checkbox for “Count Content Views”.
4. Then to implement it go to view setting in which we need it.
5. In fields add new field Total views and create a label for a better look of views.
6. Go to permission in people and give access to anonymous so that they can see total views.

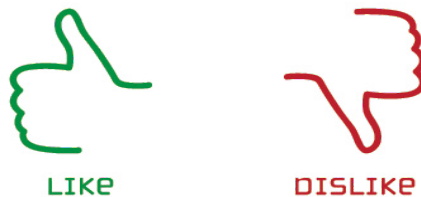


Task 2 : Implementing Like/Dislike on node

Module Used : Like/Dislike Module

This module provides "like" and "dislike" widgets for contents inside Drupal, making it easier to promote features as the one seen on many social network websites.

Technically speaking, the module provides 2 tags for Voting API, "like" and "dislike", working in a different way from Vote Up/Down, that is like a "plus or minus" approach. Likes are separate from Dislikes here.



Module needed: Like and dislike Download

Dependencies: NILL

1. Once the module is installed from drupal.org.
2. Enable the module on your Drupal site.
3. Go to the manage field of the content type add like and dislike field.
4. If you want to show it in any view then add field like and dislike in the settings of the view.
5. For the like count in the views add field like and dislike.

Task 3: Implementing Comment count on node

For comment count

Module needed: Statistics

1. The module is already installed in drupal by default.
2. Go to extend and enable the module Statistics.
3. Go to Configuration->Statistics then tick the checkbox for “Count Content Views”.
4. Then to implement it go to view setting in which we need it.
5. To add comment count add field Comment count and label it.
6. Go to permission in people and give access to anonymous so that they can see the comment.



The screenshot displays a Drupal node page for a book titled "Twinkle". The node title is "Twinkle" and it includes a description: "For kids who still believe in faires." Below the description, there are fields for "Publisher: ook Publishing Company", "Author: Various", and "Number of Pages: 80". There are also fields for "Price: 35.00" and "ISBN: 291007". A book cover image is shown on the left. To the right of the cover are social media sharing icons for Facebook, Twitter, and Google+, along with an "Unlike 1" link. Below the social media icons is a "Comments" section. A comment is visible from a user named "ajeet", posted on "Wed, 05/22/2019 - 16:24". The comment text is "nice book". Below the comment text are links for "Delete", "Edit", and "Reply". The user's profile information shows they are a "Member for 1 month 3 weeks". At the top of the node, there are buttons for "View", "Edit", "Likes", "Delete", and "Voting results". There is also an "Add new comment" link with a count of "5".

Task 4 : Implementing Share Button for Social Media Platform

Module Used : AddToAny Share Buttons Module



Share buttons for Drupal including the AddToAny universal share button, Facebook, Twitter, Google+, Pinterest, WhatsApp and many more.

Customizable

- **Documentation & Examples**
- **Your choice of share buttons with the standard buttons packaged in the module, or you can use buttons of your own**
- **Customize the SVG sharing icons by choosing a custom icon color, or using CSS code (for animations, effects, height & width, border-radius, and much more)**
- **Choose the colors of the universal share menu**
- **Fine-tune the styling of the sharing menu using CSS**

1. Click on configuration ->web services -> Add to any.
 2. Here, under entities choose the types where you want these sharing buttons.
 3. Save configuration.
 4. Now, you will notice that you can see these share buttons on article content type but not on the content type you created.
 5. So, in order to see these buttons on the content type you created.
 6. Go to that content type -> manage fields -> manage display.
 7. Manage display drag and put the addtoany inside the field of your content type(do this for all content types).
 8. Add the AddtoAny field wherever you want to put it.
 9. Save everything.
- Now, go back to your site and you will see these buttons.



Task 5 : Implementing IF Else Logic in Twig code

Module Used : Twig Field Value

Twig Field Value allows Drupal 8 themers to get partial data from field render arrays. It gives them more control over the output without drilling deep into the render array or using preprocess functions.

Single field value

```
<strong>{{ content.field_name|field_label }}</strong>: {{ content.field_name|field_value }}
```

Multiple field values

```
<strong>{{content.field_name|field_label}}</strong>: {{
content.field_name|field_value|safe_join(', ') }}
```

Single field properties

```
<span>Text format: {{ content.field_body|field_raw('text_format') }}.</span>
```

Referenced entities

```
<img src={{file_url(content.field_image|field_target_entity.uri.value) }} alt={{
content.field_image|field_raw('alt') }} />
```

```
<body
  {% if page|default('login') == 'login' %}
    class="login"
  {% elseif page == 'other' %}
    class="login"
  {% else %}
    class="noclass"
  {% endif %}>
</body>
```



Task 6 : To Explore the Taxonomy use on an instance

Create News Category vocabulary and add it to the Recipe content type as a field that can contain an unlimited number of values and that allows adding new terms to the vocabulary.

Steps


1. In the Manage administrative menu, navigate to Structure > Taxonomy(admin/structure/taxonomy).

Taxonomy

[Home](#) » [Administration](#) » [Structure](#)

Taxonomy is for categorizing content. Terms are grouped into vocabularies. For example, a vocabulary called "Fruit" would contain the terms "Apple" and "Banana".

[+ Add vocabulary](#)

VOCABULARY NAME	DESCRIPTION	OPERATIONS
Tags	Use tags to group articles on similar topics into categories.	List terms 

2. Click Add vocabulary, and fill in the values below. Like in this news category is added

[+ Add vocabulary](#)

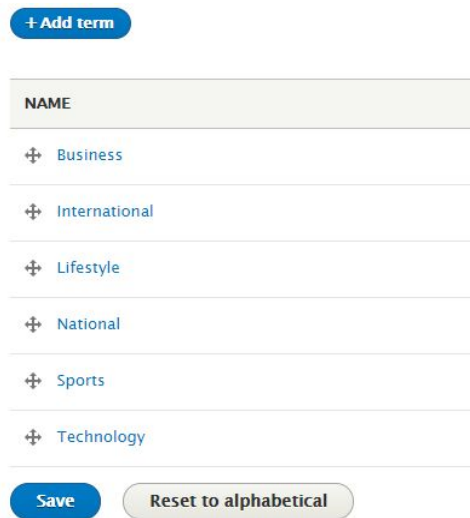
VOCABULARY NAME	DESCRIPTION
 News Category	
 Tags	Use tags to grou

[Save](#)

3. Click Save. You will be taken to the Ingredients page, which shows a list of all the terms in this vocabulary.



4. Click Add term. Terms added .



NAME
+ Business
+ International
+ Lifestyle
+ National
+ Sports
+ Technology

Save Reset to alphabetical

5. You will receive a confirmation about the term you created. Add more terms.

6. In the Manage administrative menu, navigate to Structure > Content Types(admin/structure/types).

7. Click Add field, and enter values from the table below. Click Save and continue.

8. On the following configuration screen, enter the values from the table below. Click Save field settings.

9. On the following configuration screen, enter the values from the table below. Click Save settings.

10. Click Save settings. You will be taken back to the Manage Fields page.

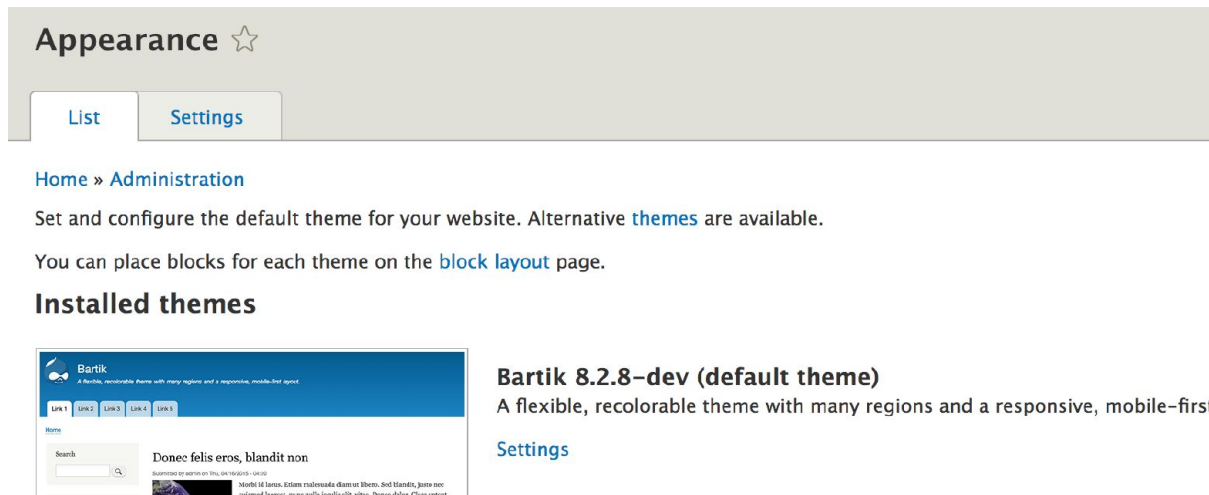
11. Final taxonomy on main navigation after implementing all the settings



Task 7 : To explore the theming of bootstrap And To create custom login page

Theming

The most obvious part of Drupal's theming system is the **Appearance** admin page found at admin/appearance, which lists all the themes installed on your website. The page is shown in the following screenshot:



Choose a theme from the **Appearance** page, you are applying a specific graphic design to your website's data and functionality. However, the applied theme is in reality only a small part of the entire theming layer.

Twig

Theme engines are responsible for doing the actual output via template files. Although previous versions of Drupal were capable of using different theme engines, one stood out and was used 99.9 percent of the time (statistic made up by me on the spot)--PHPTemplate.

This theme engine used PHP files with .tpl.php extension and contained both markup and PHP. Seasoned Drupal developers grew accustomed to this practice, but it was always more difficult for front-end developers to use and theme against. In Drupal 8, it was abandoned in favor of the Twig templating engine created by



SensioLabs (the people responsible for the Symfony project). As mentioned, theme functions were also deprecated in favor of outputting everything through a Twig file. This brought about many improvements to the theme system and quite some joy to the front-end community. For example, it improved security and readability and made it much less important to be actually versed in PHP to be able to take part in the theming of a Drupal site.

All Twig template files in Drupal 8 have .html.twig extension.

Sub-Theming

Below are instructions on how to create a Drupal Bootstrap based sub-theme. There are several different variations on how to accomplish this task, but this topic will focus on the two primarily and most common ways.

We should never modify any theme or subtheme that is packaged and released from Drupal.org, such as Drupal Bootstrap. If we do, all changes you have made will be lost once that theme is updated. Instead, you should create a sub-theme from one of the provided starterkits (this is considered a best practice). Once we've done that, we can override CSS, templates, and theme processing.

- Using the Starterkit
- Using Source Files
 - LESS
 - SASS
 - Compile
- Override Settings
- Override Templates

Using the Starterkit

The starterkit provided by this base-theme supplies the basic file structure on how to construct a proper Bootstrap based sub-theme for use with a CDN



Provider (like jsDelivr) or for use with compiling Bootstrap Framework source files.

- Copy `./themes/bootstrap/starterkits/THEMENAME` to `./themes`.
 - Rename the THEMENAME directory to a unique machine readable name. This is your sub-theme's "machine name". When referring to files inside a sub-theme, they will always start with `./themes/THEMENAME/`, where THEMENAME is the machine name of your sub-theme. They will continue to specify the full path to the file or directory inside it. For example, the primary file Drupal uses to determine if a theme exists is: `./themes/THEMENAME/THEMENAME.info.yml`.
- Rename `./themes/THEMENAME/THEMENAME.starterkit.yml` to match `./themes/THEMENAME/THEMENAME.info.yml`.
 - Open this file and change the name, description and any other properties to suit your needs. Make sure to rename the library extension name as well: `THEMENAME/framework`.
- Rename `./themes/THEMENAME/THEMENAME.libraries.yml`.
 - (Optional) If you plan on using a local precompiler (i.e. Less or Sass) then uncomment the appropriate JavaScript entries inside this file to enable the assets provided by the Bootstrap Framework.
- Rename `./themes/THEMENAME/THEMENAME.theme`.
- Rename `./themes/THEMENAME/config/schema/THEMENAME.schema.yml`
 - Open this file and rename `THEMENAME.settings:` and `'THEMETITLE settings'`
- Rename `./themes/THEMENAME/config/install/THEMENAME.settings.yml`



Twig Template naming conventions

Drupal loads templates based on certain naming conventions. This allows you to override templates by adding them to your theme and giving them specific names.

After adding a template you must rebuild the cache in order for Drupal to discover your new template.

You can debug Twig templates to figure out which templates are being used to output the markup for any given element. More about Twig debugging here.

This page lists the conventions used for the base html structure, the page, regions, blocks, nodes, fields, and other core components. (It's good to know that it is possible to create custom template name suggestions using function `hook_theme_suggestions_HOOK_alter`.)

HTML (<head> template)

Base template: `html.html.twig` (base location: `core/modules/system/templates/html.html.twig`)

The following are some examples of how you may override the base template:

1. `html--[internalviewpath].html.twig`
2. `html--node--[nodeid].html.twig`
3. `html.html.twig`

Page template

Pattern: `page--[front|internal/path].html.twig`

Base template: `page.html.twig` (base location: `core/modules/system/templates/page.html.twig`)

1. `page--node--edit.html.twig`
2. `page--node--1.html.twig`
3. `page--node.html.twig`
4. `page.html.twig`



Regions

Pattern: region--[region].html.twig

Base template: region.html.twig (base location: core/modules/system/templates/region.html.twig)

The region template is used when a page region has content, either from the Block system or functions like hook_page_top() or hook_page_bottom(). Possible region names are determined by the theme .info.yml file.

Blocks

Pattern: block--[module|-delta].html.twig

Base template: block.html.twig (base location: core/modules/block/templates/block.html.twig)

1. block--[module]--[delta].html.twig
2. block--[module].html.twig
3. block.html.twig

"module" being the name of the module and "delta", the internal id assigned to the block by the module.

Nodes

Pattern: node--[content-type|nodeid]--[viewmode].html.twig

Base template: node.html.twig (base location: core/modules/node/templates/node.html.twig)

Theme hook suggestions are made based on these factors, listed from the most specific template to the least. Drupal will use the most specific template it finds:

1. node--[nodeid]--[viewmode].html.twig
2. node--[nodeid].html.twig
3. node--[content-type]--[viewmode].html.twig
4. node--[content-type].html.twig
5. node--[viewmode].html.twig
6. node.html.twig

Taxonomy terms

Pattern: taxonomy-term--[vocabulary-machine-name|tid].html.twig

Base template: taxonomy-term.html.twig (base location: core/modules/taxonomy/templates/taxonomy-term.html.twig)



Theme hook suggestions are made based on these factors, listed from the most specific template to the least. Drupal will use the most specific template it finds:

1. taxonomy-term--[tid].html.twig
2. taxonomy-term--[vocabulary-machine-name].html.twig
3. taxonomy-term.html.twig

Fields

Pattern: field--[type|name[--content-type]|content-type].html.twig

Base template: field.html.twig (base location:
core/modules/system/templates/field.html.twig)

Theme hook suggestions are made based on these factors, listed from the most specific template to the least. Drupal will use the most specific template it finds:

1. field--node--[field-name]--[content-type].html.twig
2. field--node--[field-name].html.twig
3. field--node--[content-type].html.twig
4. field--node--[field-name].html.twig
5. field--[field-type].html.twig
6. field.html.twig

Forums

Pattern: forums--[container|topic]--forumID].html.twig

Base template: forums.html.twig (base location:
core/modules/forum/templates/forums.html.twig)

Theme hook suggestions are made based on these factors, listed from the most specific template to the least. Drupal will use the most specific template it finds:

For forum containers:

1. forums--containers--[forumid].html.twig
2. forums--[forumid].html.twig
3. forums--containers.html.twig
4. forums.html.twig

For forum topics:

1. forums--topics--[forumid].html.twig
2. forums--[forumid].html.twig
3. forums--topics.html.twig
4. forums.html.twig



Views

Patterns:

- views-view--[viewid]--[view-display-id].html.twig
- views-view--[viewid]--[view-display-type].html.twig
- views-view--[view-display-type].html.twig
- views-view--[viewid].html.twig
- views-view.html.twig



REFERENCES

1. <https://www.drupal.org>
2. <https://www.ostraining.com>
3. <https://www.valuebound.com/resources>
4. <https://training.acquia.com/>
5. <https://drupalize.me/>
6. <https://drupal-bootstrap.org/>
7. Mastering-Drupal Book
8. Drupal 8 Module Development Book

