Unit- 4

Introduction to CMS: Overview, benefits and types, WordPress Development : Theme customization, plugin development, and best practices, Drupal CMS: Configuration, module development and site building techniques

IGDTUW RESOURCE

by Pratishtha Gupta

What is CMS?

A Content Management System is a platform that helps users create, manage, and modify website content without needing to code from scratch. It's designed to reduce the technical overhead involved in building and maintaining websites.

Introduction to CMS

Content Management System (CMS)

- helps companies manage digital content. Whole teams can use these systems to create, edit, organize, and publish content.
- acts as a single place to store content and provides automated processes for collaborative digital content management and creation using built-in (or designed) workflows.
- Different privileges and responsibilities are provided to individuals based on roles. For example, authors can post and save their work, but editors can modify and publish it. Administrators can do all these things as well as grant other people permission to update or revise content.
- helps create and manage websites and website content using minimal technical overhead.
- provides an easy and cost-effective solution for content management.

- Nearly every CMS is comprised of two parts—the front end and the back end. The front end is the part the user interacts with. It's how websites are visibly structured and styled. The front end brings HTML, CSS, and JavaScript together to deliver rich, interactive content that's styled to match your company's branding.
- The back end of a CMS is the application that is used to post new content to a website. The process begins by accessing a web interface to easily add, create, and publish content to your CMS's front end. Rather than knowing HTML, CSS, and JavaScript, you create content in an interface similar to Microsoft Word. The back end then stores this content in the database and publishes it to the front end of the website.
- Together, these two systems comprise the CMS. They allow you to publish content without understanding web technologies or building your web application from the ground up.

Benefits of CMS

- Collaboration: Multiple team members can work together on content projects, and different roles can be assigned different privileges.
- User-friendly: Users don't need to learn HTML or CSS to create and publish content.
- SEO tools: CMSs can include tools and plug-ins to help improve search engine optimization (SEO).
- Workflow management: CMSs can help keep content organized with built-in workflows.
- Real-time editing: Editors can make changes to content in real-time and see how it will look when published.
- Integration: CMSs can integrate with a company's digital asset repository to centralize resources.

Benefits of CMS

Easy to Manage

Make graphic and content change with ease

SEO Friendly

Dominate search engine and sell more everyday

ZERO Dependency

No need to depend on webmasters for any update

Time Saving

Advance Features to save time & effort

Complete Control

No need to depend on webmasters for any update

Low Cost

Get your website made in your budget

Types of CMS

1. **Coupled CMS or traditional CMS.** backend and frontend are tightly integrated

- offers a fully accessible backend that connects to and modifies a website's database and publishes content to a styled front end.
- requires dedicated web hosting to run.
- Additionally, a coupled CMS will likely require that an administrator set it up and configure the system installation for ongoing use.
- WordPress is an example of a coupled CMS, as it offers a complete package for users to install, launch a website, and publish content moving forward.

2. Software-As-A-Service (SaaS) CMS

- is also a complete, end-to-end solution, but unlike coupled CMS, SaaS CMS is hosted in the cloud which means that it requires no actual setup, installation, or preconfigured web hosting.
- is an excellent solution for companies who need a straightforward web presence, as it offers all the capabilities without any of the server or web-hosting overhead.

3. Decoupled CMS

frontend and backend are separated and communicate with each other using API

- In this, the website’s presentation is “decoupled” from the back end. The delivery system sits between the presentation of the website and accesses the backend through an application programming interface (API).
- offers greater flexibility to interact with the content created in the back end.
- For example, suppose an organization wants to use its content library for a new purpose, such as mobile applications. In that case, a decoupled CMS is a good solution as it supports multiple, adaptable applications on the front end while keeping your content and information consistent in the back end.

article example

4. Headless CMS

only has backed, doesn't have built in frontend
it allows developers to build their own front-end

- has only a back-end system that accesses a database and stores content with a custom-built, front-end web application.

- offers greater flexibility than a decoupled CMS, but it also requires considerably more work than any other option.
- usually requires a developer to design, create, and connect a front-end application.
- A headless CMS is a good solution for organizations that need complete control and flexibility over how their content is accessed. It provides content storage and organizational capabilities while allowing for a custom application on the front end—whether a website, a mobile app, or some other front end.

WordPress Development

- WordPress is a free, open-source content management system (CMS) written primarily in PHP and JavaScript and It makes content creation and management very easy.
- WordPress has a wide range of **plugins** and **themes** that can help extend the functionality of a website and make it easier to build (or maintain).
- It provides ease and flexibility for non-technical users and developers alike.

General Structure

- A few key components make up most WordPress sites.

The Database

- WordPress officially supports MySQL and MariaDB as database engines. The database will store all of your site's content, configuration, and dynamic data.

WordPress Core

- This is the source code for the application, and encompasses all of the actual core functionality of the WordPress CMS itself. This can be downloaded from WordPress.org, or can come ready-installed with your hosting. You won't typically need to modify the WordPress core, as you can satisfy your customization needs using Plugins and Themes.

Plugins Plugins don't change the design, but they allow you to add more tools and functions without writing custom code.

- Plugins are pieces of code you can create yourself, or download from sources such as the WordPress Plugin repository. Plugins can introduce new functionality and behavior to your site.

Themes

A theme controls the design and layout of a website. It defines how the site looks control the visual aspects not the content

- Themes allow you to customize and control the visual aspects of your site. The Theme will determine the look and feel of the public-facing website that your users see. Like Plugins, you can create Themes yourself, or download premade ones.

Hooks

Hooks are like special spots in a website's code where you can add or change things without touching the main code itself.

- Hooks allow you to register custom functions that WordPress will utilize at specific points during the execution of code in the Core, Themes, or Plugins. Hooks come in two varieties, Actions and Filters. Actions allow you to perform a side-effect in response to one of the supported events.

Action Hooks: These let you do something extra when a certain event happens.

Filter Hooks: These let you change something before it shows up to the user.

Plugin Development

- A WordPress plugin is a PHP file with a WordPress plugin header comment. It's highly recommended that you create a directory to hold your plugin so that all of your plugin's files are neatly organized in one place

To get started creating a new plugin, follow the steps below.

1. Navigate to the WordPress installation's wp-content directory.
2. Open the plugins directory.
3. Create a new directory and name it after the plugin (e.g. plugin-name).
4. Open the new plugin's directory.
5. Create a new PHP file (it's also good to name this file after your plugin, e.g. plugin-name.php).

- Here's what the process looks like on the Unix command line:
- In the example above, vi is the name of the text editor. Use whichever editor that is comfortable for you.
- Now that you're editing your new plugin's PHP file, you'll need to add a plugin header comment. This is a specially formatted PHP block comment that contains metadata about the plugin, such as its name, author, version, license, etc. The plugin header comment must comply with the header

```
wordpress $ cd wp-content
wp-content $ cd plugins
plugins $ mkdir plugin-name
plugins $ cd plugin-name
plugin-name $ vi plugin-name.php
```

requirements, and at the very least, contain the name of the plugin.

- Only one file in the plugin's folder should have the header comment — if the plugin has multiple PHP files, only one of those files should have the header comment.
- After you save the file, you should be able to see your plugin listed in your WordPress site. Log in to your WordPress site, and click Plugins on the left navigation pane of your WordPress Admin. This page displays a listing of all the plugins your WordPress site has. Your new plugin should now be in that list.
- When WordPress loads the list of installed plugins on the Plugins page of the WordPress Admin, it searches through the plugins folder (and its sub-folders) to find PHP files with WordPress plugin header comments. If your entire plugin consists of just a single PHP file, like Hello Dolly, the file could be located directly inside the root of the plugins folder. But more commonly, plugin files will reside in their own folder, named after the plugin.

Theme Development

- To customize the visual appearance of your site, you can create a Theme. As mentioned before, WordPress offers two main types of Themes.
- Themes take the content stored by WordPress and display it in the browser. When you create a WordPress theme, you decide how that content looks and is displayed. There are many options available to you when building your theme. The biggest limit is your imagination.

Types of Themes

- WordPress supports two primary types of themes: block and classic.
- There is also a classic subtype that is called a hybrid theme, and you'll learn about it below, too. But the most important distinction is block vs. classic.
- Technically, you can even build your own theming system altogether. That's outside the scope of this handbook, but it's at least worth noting that WordPress lets you build pretty much whatever you set your mind to.

Think of them like pieces of a puzzle that you can arrange and customize.

Block Themes

Everything is created using blocks. Think of blocks as building blocks of your website.

- are the modern method of building WordPress themes.
- generally follow a standard set of conventions and are built entirely out of blocks.
- rely on HTML-based block templates that contain block markup.

his means the content and design are structured using blocks in HTML.

- Both creators and users can edit the templates in the Site Editor.
- Users can also customize global settings and styles defined by the theme’s theme.json file through the Styles interface.
- It’s also possible to export a theme directly from the Site Editor without touching any code.
- Technically, you cannot create a new theme from scratch entirely from the editor, but you can modify the templates and styles of an existing theme—in essence, creating a custom theme of your own.

Classic Themes

- use a PHP-based templating system, which is still supported in WordPress today. They are still in wide use because they were built on the theming system that was first introduced in 2005 with the launch of WordPress 1.5. There is a long and deep history of classic theming in WordPress, which continues on.
- Unlike block themes, classic themes have far fewer standards to adhere to, but there are APIs you can use for specific features.
- The classic theme creation process also requires some minimal PHP, HTML, and CSS code knowledge, at least.

Hybrid Themes

- Hybrid themes are merely classic themes that have adopted some modern block-related features, such as global settings and styles or block template parts. This is a widely agreed-upon term by the community, but it is not an “official” theme type. At the end of the day, hybrids are still classic themes.

Go through **Best Practices** on your own

From small businesses to big international companies, many use Drupal because it’s:

* Free (open-source)
* Secure
* Very customizable

Drupal CMS

- is an open-source content management system (CMS).
- gives teams the freedom to create exceptional digital experiences.
- Organizations of all sizes — across industries and the planet — use Drupal not only to build corporate websites, but to build and manage e-commerce sites, mobile applications, digital signage, social networking sites, intranets, portals, microsites, resource directories, kiosks, and more.

Drupal Configuration

control how your website works — like menus, permissions, themes, or content types.

Drupal's configuration system manages the admin settings that determine how a site functions. The configuration system allows for the following:

- Exporting, importing, and syncing
- The configuration system allows for the export, import, and syncing of configuration changes across different environments. This ensures consistency and allows for version control and easy deployment

Overriding configuration

You can temporarily change settings without fully replacing them these overrides can be saved too.

- Drupal has a configuration override system that allows for the temporary overriding of configuration values. Overrides can be stored with other configuration files for staging and version control.

Storing configuration data

- By default, Drupal stores configuration data in the database but can also be exported to YAML files.

(a readable text format), which is useful for developers.

Managing configuration

It’s like a tool to manage all the important settings for your site in an easy way.

- The Configuration Manager core module can import, export, and synchronise site configuration.

Some things that are typically included in configuration are: site name, content types and fields, taxonomy vocabularies, and views.

Here are some tips for configuring Drupal:

- Keep configuration files out of the root directory.
- In the Email Options section, fill out the name and email address to send emails from, and choose whether to allow HTML emails.

- Enter an email address to send a test email to.
- Click the Save configuration button when done.

Module Development

- The power of Drupal lies in its modules. With thousands of core and contributed Drupal modules to choose from, Drupal developers can easily implement ones that meet their needs.

Step 1: Name the Drupal 9 Module

First, we need to create a custom module under ‘web/modules/custom’ folder. We will name the folder as welcome_module.

Some things that you should keep in mind before giving a machine name to your module are:

- It should not start with uppercase letters.
- There should not be any spaces between the words.

Step 2: Get noticed with the info.yml file

We have to create a yaml file with the machine name of our module for Drupal to be able recognize the module. I’ll create a yaml file like this welcome_module.info.yml.

Here is our welcome_module.info.yml file created under "welcome" directory we created in Step 1.

- We have to create a yaml file with the machine name of our module for Drupal to be able recognize the module. I'll create a yaml file like this welcome_module.info.yml.
- Here is our welcome_module.info.yml file created under "welcome" directory we created in Step 1.

Step 3: Creating the routing file with routing.yml This file tells Drupal what URL your module should respond to.

- Next step is to add a welcome_module.routing.yml file under the "welcome" directory:
- The first line is the route name [welcome_module.my_welcome].

```
name: Welcome Module
type: module
description: 'First Custom Drupal 9 Module.'
package: Custom
version: 1.0
core_version_requirement: ^8 || ^9
```

```
welcome_module.welcome:
  path: '/welcome/page'
  defaults:
    _controller: '\Drupal\welcome_module\Controller
\WelcomeController::welcome'
    _title: 'Welcome to My Module in Drupal 9'
  requirements:
    _permission: 'access content'
```

- Under path, we specify the URL path we want this route to register. This is the URL to the route.
- Under defaults, we have two things: the _controller which references a method on the WelcomeController class and the default page title (_title).
- Under requirements, we specify the permission of the accessing. User needs to have to be able to view the page.
- Step 4: Adding a Controller
- Create a folder "modules/custom/welcome_module/src/Controller". In this folder, create a file named "WelcomeController.php" with the following content:
- Now Login to your Drupal site and enable your module. To check if it functions properly, visit the path you specified in your routing file, that is /welcome/page. If you get the ‘Page Not Found’ error, then

```
<?php
namespace Drupal\welcome_module\Controller;
class WelcomeController {
  public function welcome() {
    return array(
      '#markup' =----> 'Welcome to our Website.'
    );
  }
}
```

clear the cache by navigating to admin->configuration->performance. Check again and it should function properly this time.

Site Building Techniques

Here are some techniques for building a site with Drupal:

- **Select a theme and layout:** The first impression of a website is important, so choose a theme and layout that is attractive, responsive, and easy to customize.
- **Add content:** Create content types, fields, and taxonomy vocabularies.
- **Design navigation:** Use menus to design the site's navigation.
- **Set up access control:** Use roles and permissions to control access.
- **Manage configuration:** Sync and manage the site's configuration.
- **Set up workflows:** Set up content moderation and workflows.

- **Optimize performance:** Use caching to speed up the site's performance. You can use page caching, twig caching, and BigPipe caching.
- **Use command line tools:** Learn to use command line tools like Composer, Git, and Drush.
- **Install modules:** Add, update, and remove contributed modules using Composer.
- **Configure system settings:** Configure system and account settings.
- **Understand site reports:** Understand site reports and how to initiate and update .php.
- **Configure error messages:** Configure error messages and logging.