В. І. Юскович-Жуковська, О. М. Богут

ОСНОВИ ВЕБ-ПРОГРАМУВАННЯ

ПІДРУЧНИК ДЛЯ ВНЗ

РІВНЕ - 2019

Цей підручник з веб-програмування на PHP надає читачам комплексне розуміння мови програмування PHP та її використання в розробці веб-додатків. Ви дізнаєтесь про основні концепції, синтаксис та функціональні можливості PHP, які дозволять вам створювати динамічні та інтерактивні веб-сторінки.

Починаючи з основ, ви зазначите крок за кроком процес розробки на PHP, включаючи роботу з змінними, масивами, функціями та умовними операторами. Ви дослідите роботу з формами та взаємодію з базами даних, навчитеся створювати запити та обробляти результати.

Підручник також пропонує огляд фреймворків та інструментів, які сприяють розробці на PHP, таких як Drupal. Ви дізнаєтеся, як використовувати ці фреймворк для створення потужних та масштабованих веб-додатків.

Крім того, ви будете ознайомлені з кращими практиками розробки на PHP, включаючи забезпечення безпеки, оптимізацію продуктивності та розгортання веб-додатків на різних серверах.

Цей підручник призначений для початківців та тих, хто вже має деяке базове розуміння веб-розробки. Він надає чіткі пояснення, приклади коду та практичні завдання, що допоможуть вам освоїти PHP та використовувати його для розробки веб-додатків з впевненістю.

ВСТУП

Ласкаво просимо до підручника з основ веб-програмування! Цей підручник призначений для всіх, хто бажає вивчити основи веб-програмування та розробки веб-додатків. Ми дослідимо ключові мови та технології, такі як HTML, CSS, PHP, а також розглянемо сучасну роль веб-програмування в суспільстві сьогодні.

Сьогодні веб-програмування має величезне значення в різних сферах, таких як електронна комерція, соціальні мережі, освіта, медіа та багато інших. Завдяки йому ми можемо створювати потужні, інтерактивні та відзивчиві додатки, які забезпечують зручну та змістовну взаємодію з користувачем.

У цьому підручнику ми ознайомимося з основами веб-програмування, починаючи з важливих концепцій HTML, CSS та PHP. Ми розглянемо структуру веб-сторінок, властивості стилів, роботу з формами, роботу з базами даних та багато іншого, а також розглянемо використання фреймворку Drupal.

Drupal 8 є одним із найпопулярніших веб-фреймворків у світі веб-розробки. Він володіє широким спектром можливостей і забезпечує зручний інтерфейс для створення та управління веб-сайтами. Цей фреймворк дозволяє розробникам ефективно працювати з базою даних, модулями, темами, розширеннями та іншими компонентами Drupal, що робить його ідеальним вибором для будь-якого проекту веб-розробки.

У цьому підручнику ми покриємо основи роботи з Drupal 8, починаючи зі створення структури сайту та налаштування його компонентів. Ви дізнаєтеся, як створювати та керувати контентом, використовувати модулі для розширення функціональності, налаштовувати теми для оформлення зовнішнього вигляду вашого сайту, працювати з базою даних та виконувати інші завдання, пов'язані з розробкою веб-додатків.

Незалежно від вашого рівня досвіду веб-програмування, цей підручник надасть вам чітке уявлення про процес розробки на Drupal 8. Ми почнемо з основ, а потім перейдемо до більш складних тем, дозволяючи вам розширити свої знання та навички.

Ми рекомендуємо вам активно виконувати практичні завдання та вправи, що наведені в цьому підручнику, оскільки це дозволить вам закріпити набуті знання та отримати практичний досвід роботи з Drupal 8. Нехай цей підручник стане вашим провідником у захоплюючому світі веб-програмування. Почнемо наше навчання та покладемо основи для вашої успішної кар'єри в цій галузі!

Приємного навчання!

ГЛОСАРІЙ

HTML (HyperText Markup Language) - мова розмітки, використовується для створення структури та семантики веб-сторінок.

CSS (Cascading Style Sheets) - мова стилів, використовується для оформлення веб-сторінок та керування їх зовнішнім виглядом.

PHP (Hypertext Preprocessor) - мова програмування, призначена для розробки динамічних веб-сторінок та взаємодії з базою даних.

Тег - елемент HTML, який використовується для визначення різних частин веб-сторінки.

Атрибут - значення, яке використовується для налаштування або додаткової інформації про тег HTML.

Селектор - частина CSS, яка вказує, які елементи HTML будуть застосовуватись до заданого стилю.

Клас - атрибут HTML та селектор CSS, які використовуються для ідентифікації групи елементів з однаковими стилевими правилами.

ID - атрибут HTML, який ідентифікує унікальний елемент на сторінці та дозволяє застосовувати до нього специфічні стилі.

Розмітка - структура та організація елементів на веб-сторінці за допомогою HTML.

Блоковий елемент - HTML-елемент, який займає всю доступну ширину сторінки та може містити інші елементи.

Інлайновий елемент - HTML-елемент, який займає тільки стільки простору, скільки необхідно для відображення свого вмісту.

Заголовок - HTML-тег, що використовується для визначення заголовків різних рівнів на веб-сторінці.

CSS-селектори - патерни, які використовуються для вибору конкретних елементів HTML для застосування стилів.

Відступ - відстань між внутрішнім вмістом елемента та його зовнішніми межами.

Властивість - параметр CSS, який визначає певний аспект стилю, такий як колір, розмір шрифту, відстань тощо.

Інтерактивність - здатність веб-сторінки реагувати на дії користувача, такі як кліки, наведення курсору тощо.

JavaScript - мова програмування, яка використовується для створення динамічного та інтерактивного веб-контенту.

Функція - блок коду, який виконує певні дії або обчислює значення, що може бути викликаний з інших частин програми.
Змінна - контейнер для зберігання значення, яке може змінюватися під час виконання програми.
Масив - структура даних, що містить послідовність елементів, до яких можна звертатися за допомогою індексів.
Форма - елемент HTML, який дозволяє користувачам вводити та відправляти дані на сервер.
Метод передачі даних - спосіб передачі даних з форми на сервер, такі як GET або POST.
База даних - структуроване сховище даних, що використовується для зберігання та організації інформації.
SQL (Structured Query Language) - мова запитів, використовувана для взаємодії з базами даних.
Запит - команда SQL, яка виконується на базі даних для отримання, оновлення, вставки або видалення даних.
Фреймворк - набір готових компонентів, бібліотек та правил, які спрощують розробку веб-додатків.
Розширення - додатковий модуль або компонент, який додає функціональність до певного програмного забезпечення, такого як PHP.
API (Application Programming Interface) - набір правил та інструкцій, які дозволяють взаємодіяти між різними програмами та сервісами.
Респонсивний дизайн - підхід до розробки веб-сайтів, що забезпечує їх адаптацію до різних розмірів екранів та пристроїв.
Валідація - процес перевірки введених даних користувачем на відповідність заданим критеріям або формату.

Цей глосарій містить основні терміни та поняття, які будуть використовуватись протягом підручника. Вони допоможуть вам розуміти та використовувати правильну термінологію під час навчання веб-програмування на HTML, CSS та PHP.

1. ЗАГАЛЬНІ ПОНЯТТЯ ВЕБ-ПРОГРАМУВАННЯ

1.1. ПОНЯТТЯ ПРО ВЕБ-ПРОГРАМУВАННЯ. БЕКЕНД ТА ФРОНТЕНД РОЗРОБКА

Веб-програмування - це процес розробки та створення динамічних веб-додатків та веб-сайтів, які взаємодіють з користувачами через Інтернет. Веб-програмування включає в себе використання різних мов програмування, технологій та фреймворків для створення функціональних та ефективних веб-додатків.

У веб-програмуванні ключову роль відіграють три основні мови та технології:

- HTML (HyperText Markup Language) - це мова розмітки, використовується для визначення структури та семантики веб-сторінок. HTML елементи визначають різні частини сторінки, такі як заголовки, параграфи, списки, таблиці, зображення тощо.
- CSS (Cascading Style Sheets) - це мова стилів, використовується для оформлення веб-сторінок та керування їх зовнішнім виглядом. CSS визначає кольори, шрифти, розміри, відступи, рамки та інші стилеві властивості елементів на сторінці.
- JavaScript - це мова програмування, яка використовується для створення динамічного та інтерактивного веб-контенту. JavaScript дозволяє додавати взаємодію на сторінці, перевіряти дані, валідувати форми, анімувати елементи, отримувати та відправляти дані до сервера без перезавантаження сторінки.

Окрім цих основних мов, веб-програмування включає в себе використання баз даних, таких як MySQL або PostgreSQL, для збереження та організації даних, а також використання серверних мов програмування, таких як PHP, Python або Ruby, для обробки запитів та взаємодії з базою даних.

Одним з популярних фреймворків для веб-програмування є Drupal. Drupal - це відкрите програмне забезпечення для управління вмістом та розробки веб-сайтів. Він надає розширюваність, гнучкість та широкі можливості для створення складних та потужних веб-додатків.

Веб-програмування використовується для розробки різноманітних веб-додатків, таких як електронні комерційні сайти, соціальні мережі, блоги, форуми, управління вмістом та інші. Воно дозволяє створювати веб-додатки, які забезпечують користувачів інтерактивними функціями, зручним інтерфейсом та швидким доступом до необхідної інформації.

Веб-програмування використовується всюди в Інтернеті і відіграє важливу роль у сучасному світі, де веб-додатки є необхідною частиною бізнесу, комунікації та розваг.

У веб-програмуванні виділяють дві основних складових. Бекенд і фронтенд - це дві ключові складові веб-розробки, які відповідають за різні аспекти створення веб-додатків. Ось їх особливості:

Фронтенд розробка:

- Фронтенд розробка відповідає за створення того, що користувач бачить і взаємодіє на веб-сторінці.
- Вона займається розробкою користувацького інтерфейсу (UI), який включає в себе структуру, дизайн, оформлення та інтерактивні елементи веб-сторінок.
- Фронтенд розробник використовує мови та технології, такі як HTML, CSS та JavaScript, для створення зовнішнього вигляду та поведінки веб-сторінок.
- Основна мета фронтенд розробки - забезпечити зручний та естетичний користувацький досвід, який включає в себе навігацію, взаємодію з елементами, анімацію та відповідність веб-сторінок різним пристроям та браузерам.

Бекенд розробка:

- Бекенд розробка відповідає за обробку логіки, обмін даними та управління серверною стороною веб-додатка.
- Вона зосереджена на розробці серверного програмного забезпечення, яке працює на сервері та взаємодіє з базами даних, зовнішніми сервісами та іншими компонентами.
- Бекенд розробник використовує мови програмування, такі як PHP, Python, Ruby або Java, для створення функціональності та обробки запитів користувачів.
- Основна мета бекенд розробки - забезпечити надійну та ефективну роботу веб-додатка, включаючи обробку даних, безпеку, автентифікацію, маршрутизацію, керування сесіями та інші аспекти серверної логіки.

Одна з ключових особливостей роботи фронтенд та бекенд розробників полягає в тому, що вони співпрацюють між собою для створення повноцінного веб-додатка. Фронтенд розробник отримує від бекенд розробника необхідні дані та функціонал для відображення на веб-сторінці, тоді як бекенд розробник забезпечує необхідні API та

логіку для взаємодії з фронтендом. Ці дві складові взаємодіють між собою через HTTP протокол, де фронтенд надсилає запити на бекенд, а бекенд повертає необхідні дані та результати обробки.

Веб-розробка вимагає співпраці між фронтенд та бекенд розробниками, щоб створити функціональний, естетичний та ефективний веб-додаток, який забезпечує приємний користувацький досвід та відповідає потребам користувачів.

Фронтент і бекенд взаємодіють між собою в рамках технології “клієнт-сервер”.

Технологія клієнт-сервер - це архітектурний підхід до побудови розподілених комп'ютерних систем, де функції і завдання розділені між двома основними компонентами: клієнтом і сервером.

У такій системі клієнт - це кінцевий користувач або програмне забезпечення, що взаємодіє з користувачем. Клієнт зазвичай надає інтерфейс користувача, через який він взаємодіє з додатком або отримує доступ до ресурсів, що зберігаються на сервері.

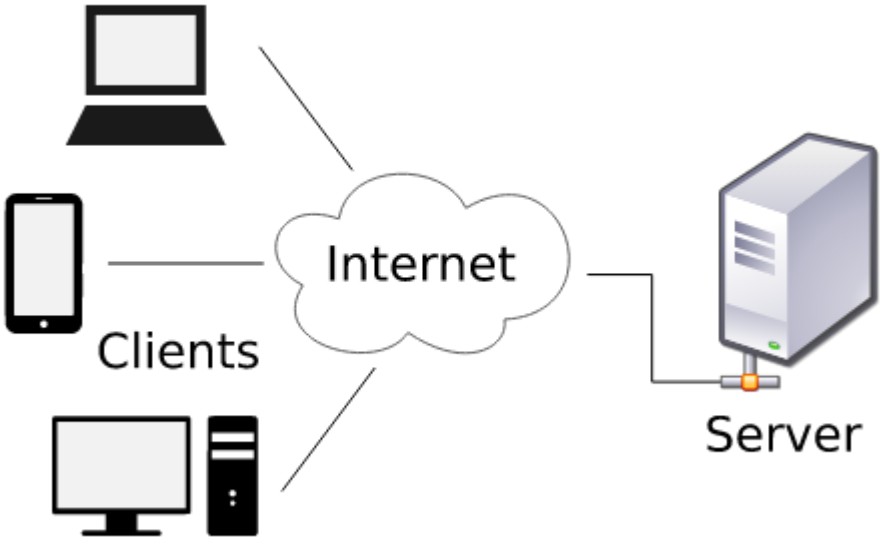

Рис. 1. Архітектура “клієнт-сервер”.

Сервер - це централізований компонент, який забезпечує доступ до ресурсів, обробляє запити клієнтів і надає їм необхідну інформацію або послуги. Сервер може бути фізичним комп'ютером або набором комп'ютерів, що працюють разом у мережі.

Основна ідея технології клієнт-сервер полягає в тому, що обробка даних та виконання операцій розподіляється між клієнтом і сервером. Клієнтська сторона відповідає за представлення даних і взаємодію з користувачем, тоді як серверна сторона забезпечує обробку даних, збереження інформації та відповіді на запити

клієнтів. Ця модель дозволяє розділити навантаження між клієнтами і серверами, полегшує розширення системи, спрощує підтримку і оновлення програмного забезпечення. Технологія клієнт-сервер є основою для багатьох розподілених додатків та мережевих послуг, таких як веб-сайти, електронна пошта, системи управління базами даних, хмарні обчислення та багато іншого.

Зазвичай, взаємодія між клієнтом та сервером, у випадку веб-додатків, здійснюється за протоколом HTTPS. Протокол HTTPS (Hypertext Transfer Protocol Secure) є захищеним протоколом передачі гіпертексту, який використовується для безпечного обміну інформацією між веб-браузером користувача і веб-сервером. Він є шифрованим варіантом стандартного протоколу HTTP і забезпечує конфіденційність та цілісність даних, переданих між клієнтом і сервером. У протоколі HTTPS використовується шифрування для захисту передачі даних. Це досягається за допомогою використання криптографічних протоколів, таких як SSL (Secure Sockets Layer) або його сучасного наступника TLS (Transport Layer Security). Ці протоколи забезпечують шифрування даних, що передаються між клієнтом і сервером, тим самим унеможливлюючи прослуховування та підробку інформації третіми сторонами. Коли користувач взаємодіє з веб-сайтом за допомогою HTTPS, його веб-браузер встановлює безпечне з'єднання з сервером, а потім застосовує шифрування для кожної передачі даних. Це включає в себе валідацію сертифіката сервера, що підтверджує його автентичність.

Використання HTTPS має декілька переваг. Воно забезпечує конфіденційність, захищаючи дані від прослуховування. Воно також забезпечує цілісність, що означає, що дані не можуть бути змінені під час передачі. Крім того, використання HTTPS допомагає підтримувати довіру користувачів до веб-сайту, оскільки вони бачать піктограму замка або індикатор безпеки в адресному рядку браузера.

У сучасному Інтернеті використання HTTPS є широко поширеним і є стандартним для більшості веб-сайтів, особливо тих, які обробляють чутливу особисту інформацію, таку як паролі, фінансові дані або особисті дані користувачів.

Питання для самоконтролю:

1. Які мови та технології використовуються у веб-програмуванні?
2. Що таке фронтенд та бекенд?
3. Що таке технологія клієнт-сервер?
4. Для чого використовується протокол HTTPS?

1.2. КЛАСИФІКАЦІЯ МОВ ПРОГРАМУВАННЯ ТА РОЗМІТКИ

Перед тим, як перейти до огляду мов, які використовуються у веб-програмуванні, давайте визначимо принципову різницю між мовами розмітки та програмування.

Мова розмітки і мови програмування є різними концепціями, які використовуються для різних цілей у сфері комп'ютерних наук. Ось декілька ключових відмінностей між ними:

Мова розмітки:

- Мова розмітки (наприклад, HTML, XML) використовується для описування структури, форматування та презентації документів.
- Вона використовується для розмітки контенту з метою його відображення в браузері або інших програмах.
- Мови розмітки не є програмувальними, оскільки вони не мають повноцінного набору функцій і можливостей для виконання складних операцій.
- Вони визначають структуру документа, вказують як різні елементи пов'язані між собою, а також задають способи відображення контенту.

Мова програмування:

- Мова програмування (наприклад, Python, Java, C++) використовується для створення програм і алгоритмів, які виконують обчислення і керують роботою комп'ютера.
- Вона надає засоби для створення програм з різноманітним функціоналом, включаючи обробку даних, взаємодію з користувачем, маніпулювання файлами та інші операції.
- Мови програмування мають повноцінний синтаксис, набір побудованих функцій, можливості створення змінних, виконання умовних операторів, циклів та інші засоби для реалізації алгоритмів.
- Вони використовуються для розробки програмних додатків, веб-сайтів, ігор, мобільних додатків та багатьох інших систем.

У певних випадках мови розмітки і мови програмування можуть взаємодіяти між собою. Наприклад, у веб-розробці мови розмітки (наприклад, HTML) використовуються для створення структури веб-сторінок, а мови програмування (наприклад, JavaScript) використовуються для додавання динамічного функціоналу і взаємодії з користувачем на веб-сторінці.

Отже, хоча мови розмітки і мови програмування подібні за певних умов, їхній основний призначення і функціональні можливості різняться.

Давайте розглянемо особливості мов, які надалі будемо використовувати

1.2.1. МОВА HTML ТА ЇЇ ОСОБЛИВОСТІ

Мова HTML (Hypertext Markup Language) є стандартною мовою розмітки, що використовується для створення структури та вмісту веб-сторінок. HTML дозволяє визначати, як елементи веб-сторінки пов'язані між собою і як вони повинні бути відображені в браузері.

Основна концепція HTML полягає в тому, що документ поділяється на різні елементи, які описують структуру сторінки. Кожен елемент має свою властивість і містить певний контент або інші вкладені елементи. Це дозволяє визначати заголовки, параграфи, списки, таблиці, зображення, посилання та інші елементи, які створюють вміст веб-сторінки.

HTML використовує ряд тегів для опису структури документа. Теги мають вигляд <назва_тегу>, і вони використовуються для позначення початку і кінця елемента. Наприклад, ... визначає параграф, <h1>...</h1> визначає заголовок першого рівня, <img> вставляє зображення і т.д.

Крім того, HTML також дозволяє використовувати атрибути для налаштування властивостей елементів, таких як класи, ідентифікатори, стилі, посилання та інші. Атрибути вказуються всередині відповідного тегу, наприклад: посилання.

HTML є основою для веб-розробки і використовується разом з CSS (Cascading Style Sheets) і JavaScript для створення багатофункціональних веб-сторінок і веб-додатків. CSS використовується для визначення стилів і вигляду елементів, а JavaScript додає динамічний функціонал і інтерактивність до веб-сторінок.

HTML документ має певну структуру, яка визначає організацію елементів і їх взаємозв'язок в межах документа. Основна структура HTML документа складається з наступних частин:

Тип документа (Document Type Declaration): Це перша лінія в HTML файлі, що вказує браузеру на версію HTML, яку слід використовувати. Наприклад, `<!DOCTYPE html>` показує, що файл є HTML5 документом.

Елемент `<html>`: Весь зміст HTML документа поміщений всередині елемента `<html>`. Він визначає кореневий елемент документа.

Елемент `<head>`: Елемент `<head>` містить метадані документа, такі як заголовок сторінки, підключення зовнішніх файлів CSS або JavaScript, метатеги для пошукових систем, інформація про кодування тощо. Цей вміст не відображається на веб-сторінці, але надає важливу інформацію про документ.

Елемент `<body>`: Елемент `<body>` містить весь вміст веб-сторінки, який буде відображений в браузері. Він може містити заголовки, параграфи, зображення, посилання, таблиці, форми та інші елементи, що складають зміст сторінки.

Ось приклад структури HTML документа:

```
html
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Заголовок сторінки</title>
  <!-- Підключення зовнішніх файлів CSS або JavaScript -->
  <link rel="stylesheet" href="styles.css">
  <script src="script.js"></script>
</head>
<body>
  <!-- Вміст веб-сторінки -->
  <h1>Заголовок першого рівня</h1>
  <p>Це параграф з текстом.</p>
  <img src="image.jpg" alt="Зображення">
  <a href="https://www.example.com">Посилання</a>
  <!-- Та інші елементи -->
</body>
</html>
```

Заголовок сторінки, підключення зовнішніх файлів, а також вміст сторінки розміщуються відповідно всередині `<head>` і `<body>`. Ця структура дозволяє браузеру правильно інтерпретувати та відображати веб-сторінку згідно з заданими налаштуваннями.

Мова HTML має два основних стандарти, які використовуються в сучасному веб-програмуванні. HTML4 і HTML5 є двома версіями мови HTML, які використовуються для створення веб-сторінок. Зробимо короткий огляд кожної з версій:

HTML4 був стандартом, прийнятим у 1997 році. Він вніс багато основних можливостей, які використовуються до сьогодні. Основні риси HTML4 включають:

- Строгі теги: HTML4 використовує строгу структуру тегів, що обмежує можливості маркування та стилізації елементів.
- Визначення таблиць: HTML4 дозволяє створювати таблиці для організації даних з використанням елементів `<table>`, `<tr>`, `<td>` та ін.
- Фрейми (Frames): HTML4 підтримує використання фреймів, що дозволяють розділити веб-сторінку на кілька незалежних областей, які можуть завантажувати окремі документи.
- Форми та інтерактивні елементи: HTML4 надає елементи форми, такі як `<input>`, `<select>`, `<textarea>`, для створення взаємодії з користувачем.

HTML5 є останньою версією HTML, що була офіційно прийнята в 2014 році. Вона внесла значні поліпшення та нові можливості. Основні особливості HTML5 включають:

- Розширена семантика: HTML5 надає нові семантичні елементи, такі як `<header>`, `<nav>`, `<footer>`, `<article>`, які полегшують розуміння структури веб-сторінки для браузерів і пошукових систем.
- Мультимедіа: HTML5 включає підтримку нативного відтворення аудіо та відео без використання плагінів, за допомогою елементів `<audio>` і `<video>`.
- Графіка і візуалізація: HTML5 включає можливості для малювання векторної графіки за допомогою тега `<canvas>`, а також для створення складних візуалізацій та анімацій з використанням технологій, таких як SVG (Scalable Vector Graphics).

- Локальне сховище: HTML5 має можливості локального сховища (Local Storage), що дозволяє зберігати дані на боці клієнта для підтримки офлайн режиму та збереження стану додатків.
- Покращена форма і валідація: HTML5 включає нові типи полів вводу, такі як елементи для вибору дати, номерів телефону, електронної пошти тощо. Він також має вбудовані засоби для валідації форм на боці клієнта.

HTML5 також включає інші функції, такі як геолокація, веб-роботи, покращена підтримка мобільних пристроїв тощо. Враховуючи свої розширені можливості, HTML5 став важливим стандартом для сучасної веб-розробки.

Важливим елементом, на який варто звертати увагу є семантика. Семантика HTML відноситься до використання тегів і елементів HTML з метою визначення значення та смислу вмісту веб-сторінки. Замість того, щоб використовувати теги лише для форматування та вигляду, семантичне використання HTML дозволяє браузерам, пошуковим системам та іншим інструментам краще розуміти структуру сторінки та її зміст.

Основні принципи семантичного використання HTML включають:

1. Використання семантичних елементів: HTML5 введе багато нових семантичних елементів, таких як `<header>`, `<nav>`, `<footer>`, `<article>`, `<section>`, `<aside>`, `<main>` і т.д. Ці елементи надають контекст і семантику для різних частин веб-сторінки, допомагають розуміти їхню роль та взаємозв'язок.
2. Відповідність використанню тегів: Використовуйте теги, які найкраще відповідають типу вмісту, який ви розміщуєте. Наприклад, використовуйте `<h1>`-`<h6>` для заголовків, `<p>` для абзаців тексту, `<ul>` і `<ol>` для списків, `<img>` для зображень і т.д.
3. Семантичні атрибути: HTML надає семантичні атрибути, які допомагають додатково визначати семантику елементів. Наприклад, `alt` атрибут для `<img>` описує альтернативний текст для зображення, `href` атрибут для `<a>` визначає адресу посилання, а `placeholder` атрибут для `<input>` надає підказку у полі вводу.
4. Заголовки та структура документа: Використовуйте заголовки `<h1>`-`<h6>` для ієрархічного визначення структури сторінки. Заголовки допомагають розуміти важливість різних розділів і покращують доступність сторінки.

Обсяг підручника не дозволяє надати повноцінну ступінчату програму вивчення HTML, але все ж наведемо основні рекомендції.

При вивченні HTML рекомендуємо звертати увагу на наступні аспекти:

1. Синтаксис та структура: Ознайомтесь з основними правилами синтаксису HTML, включаючи використання тегів, атрибутів, знаків розділювачів і правильного вкладення елементів. Розуміння правильної структури документа допоможе вам будувати правильні HTML сторінки.
2. Семантика: Вивчайте семантичне використання HTML, використовуючи семантичні теги і атрибути, які надають змісту веб-сторінки його значення і структуру. Розуміння семантики допоможе покращити доступність, SEO і зрозумілість ваших веб-сторінок.
3. Елементи форми: Детально вивчайте елементи форми HTML, такі як `<input>`, `<select>`, `<textarea>`, `<label>`, а також атрибути, які дозволяють виконувати різні завдання збору інформації від користувачів. Форми є важливою частиною багатьох веб-сторінок і веб-додатків.
4. Зображення та мультимедіа: Розберіться з використанням тегу `<img>` для вставки зображень, атрибуту `alt` для альтернативного тексту зображення. Досліджуйте можливості відео та аудіо в HTML з використанням тегів `<video>` і `<audio>`.
5. Стилізація з CSS: Розуміння взаємодії HTML з CSS (Cascading Style Sheets) допоможе вам зрозуміти, як впливати на вигляд і форматування веб-сторінок. Вивчайте базові концепції CSS, включаючи вибір селекторів, властивості стилів, класи та ідентифікатори.
6. Ресурси та документація: Використовуйте доступні онлайн-ресурси, документацію та посібники для отримання додаткової інформ

Важливо також пам’ятати про кросбраузерність. Кросбраузерна розмітка (або кросбраузерна сумісність) відноситься до практики створення веб-сторінок або веб-додатків, які однаково коректно відображаються та працюють на різних веб-браузерах. Оскільки різні браузери можуть інтерпретувати HTML та CSS різними способами, кросбраузерна розмітка є важливою для забезпечення однакового вигляду та функціональності сторінок незалежно від веб-браузера, яким користувачі користуються.

Основні аспекти кросбраузерної розмітки включають:

1. Правильне використання синтаксису та стандартів: Важливо дотримуватися правильного синтаксису HTML та CSS згідно зі стандартами, такими як HTML5 і CSS3. Це допоможе забезпечити більшу сумісність між різними браузерами.
2. Тестування на різних браузерах: Перед публікацією веб-сторінки або веб-додатку рекомендується тестувати його на різних веб-браузерах, таких як Google Chrome, Mozilla Firefox, Safari, Microsoft Edge та інші. Це дозволяє виявити та виправити проблеми, що виникають через різні інтерпретації.
3. Поліфіли та використання веб-шрифтів: Використання поліфілів (polyfills) - це техніка, яка дозволяє емулювати певні можливості HTML5 та CSS3 на старіших браузерах, які не підтримують ці можливості. Використання веб-шрифтів також може забезпечити однаковий вигляд тексту на різних браузерах.
4. Гнучкий дизайн (Responsive design): Забезпечення адаптивності веб-сторінки на різних пристроя

1.2.2. МОВА CSS ТА ЇЇ ОСОБЛИВОСТІ

CSS (Cascading Style Sheets) є мовою розмітки, яка використовується для опису вигляду та стилізації елементів HTML і XML документів. Вона дозволяє веб-розробникам контролювати вигляд, розташування, кольори, шрифти та інші аспекти веб-елементів, щоб зробити їх привабливими та зручними для користувачів.

Основні поняття CSS включають:

1. Селектори: Селектори визначають, які елементи HTML або XML будуть стилізовані. Вони можуть базуватися на класах, ідентифікаторах, типах елементів або їхній ієрархії. Наприклад, `h1` визначає стилі для всіх заголовків першого рівня `<h1>`, `.class` визначає стилі для елементів з вказаним класом, `#id` визначає стилі для елемента з вказаним ідентифікатором.
2. Властивості: Властивості визначають різні аспекти вигляду елементів. Вони можуть контролювати розміри, кольори, шрифти, відступи, фон, рамки та багато іншого. Наприклад, `color` задає колір тексту, `font-size` встановлює розмір шрифту, `background-color` визначає колір фону елемента.

3. Значення: Кожна властивість має свої значення, що визначають, яким чином вона буде застосовуватись до елементів. Наприклад, значення `red` для `color` задає червоний колір тексту, значення `12px` для `font-size` встановлює розмір шрифту 12 пікселів.
4. Каскадність та спадковість: CSS має принцип каскадності, що означає, що стилі можуть бути визначені на різних рівнях і вони будуть спадати на дочірні елементи. Наприклад, стилі, визначені на рівні документа, будуть застосовуватись до всіх його елементів, але можуть бути перевизначені на рівні окремих елементів або класів.
5. Розташування елементів: CSS надає засоби для контролю над розташуванням елементів на сторінці, такі як відступи, межі, плавання, позиціонування та гнучкій бокс.
6. Медіазапити: CSS також підтримує медіазапити, які дозволяють застосовувати стилі в залежності від параметрів пристрою або екрану, на якому відображається сторінка. Це дозволяє створювати адаптивні та мобільно-дружні веб-сторінки.

Застосування CSS допомагає зробити веб-сторінки більш естетично привабливими, зручними для використання та легкими для підтримки на різних браузерах і пристроях.

Актуальна версія CSS - це CSS3. CSS3 відноситься до третьої версії Cascading Style Sheets (CSS), яка є набором розширень та нових можливостей, які були внесені в CSS з метою поліпшення стилізації та вигляду веб-сторінок. CSS3 включає багато нових функцій і модулів, що розширюють можливості CSS, дозволяючи веб-розробникам створювати більш динамічні та привабливі веб-сайти.

Основні можливості CSS3 включають:

- Радіуси кути: CSS3 дозволяє встановлювати закруглені кути для блокових елементів з використанням властивостей `border-radius` і `border-radius`, що дають більш м'який і сучасний вигляд.
- Тіні: За допомогою властивості `box-shadow` можна додавати тінь до елементів, створюючи ефекти тінювання і виділення.
- Переходи та анімація: CSS3 надає можливість створювати плавні переходи між станами елементів з використанням властивостей `transition` та `animation`. Це дозволяє створювати рухливі ефекти, які покращують взаємодію з користувачем.

- Градієнти: CSS3 дозволяє створювати градієнтні фони за допомогою властивості `linear-gradient` або `radial-gradient`, що дозволяє створювати багатофарбові і гармонійні фонові ефекти.
- Перетворення: CSS3 надає можливості перетворення елементів, такі як масштабування, поворот, нахил і зсув. Властивості `transform` дозволяють виконувати різні операції з елементами.
- Гнучкі розміри і медіа-запити: CSS3 дозволяє створювати адаптивні та гнучкі розміри елементів за допомогою властивостей `flexbox` і `grid`. Також він підтримує медіа-запити, що дозволяють застосовувати різні стилі в залежності від розміру екрану або пристрою.

Це лише кілька з основних можливостей CSS3. Всього CSS3 включає багато нових модулів, які надають широкі можливості для стилізації та вигляду веб-сторінок.

Як ми вже згадували вище - в сучасних підходах до верстки використовуються два основних концепти - гріди та флексбокс.

Гріди (CSS Grid) і флексбокси (Flexbox) - це дві різні технології, що використовуються для розміщення та організації елементів на веб-сторінках. Ось основні відмінності між ними:

Орієнтація:

- Флексбокс: Флексбокс працює в одновимірному просторі, тобто здатний управляти розташуванням елементів лише в одному напрямку - горизонтальному або вертикальному.
- Гріди: Гріди працюють в двовимірному просторі, що дозволяє управляти розміщенням елементів як горизонтально, так і вертикально, формуючи сітку з рядків і колонок.

Структура:

- Флексбокс: Флексбокс має гнучку структуру, що дозволяє змінювати розміри та порядок елементів в залежності від доступного простору.
- Гріди: Гріди мають більші можливості щодо складних макетів, де можна визначати розташування елементів в конкретних рядках та колонках.

Управління простором:

- Флексбокс: Флексбокс надає досить простий спосіб управління розміщенням елементів та розподілом вільного простору між ними, застосовуючи властивості, такі як `justify-content`, `align-items` та `flex`.

- Гріди: Гріди дозволяють точно контролювати розміщення елементів вздовж рядків і колонок, використовуючи властивості, такі як `grid-template-columns`, `grid-template-rows` та `grid-area`.

Обидва підходи мають свої переваги та використовуються відповідно до конкретних потреб та вимог дизайну. Флексбокс підходить для простих одновимірних розташувань елементів, тоді як гріди є потужним інструментом для створення складних багатовимірних макетів. Часто їх використовують в поєднанні, доповнюючи один одного для досягнення потрібного розміщення та організації елементів на сторінці.

Для зручності ведення розробки використовують препроцесори, які спрощують процес створення стилів.

LESS (Leaner CSS) і SASS (Syntactically Awesome StyleSheets) є препроцесорами CSS, що надають розширені можливості для написання стилів веб-сторінок. Обидва препроцесори додають додаткові функції і функціональність до стандартного CSS, що полегшує розробку та підтримку стилів.

Основні особливості LESS/SASS та SCSS:

1. Змінні: Можна використовувати змінні для збереження значень, таких як кольори, розміри шрифту, відступи та інші, та використовувати їх в усіх стилях. Це дозволяє легко змінювати значення змінних і автоматично оновлювати стилі на всіх місцях, де використовуються ці змінні.
2. Вкладеність: Можна вкладати стилі одного елемента в стилі іншого елемента. Це допомагає структурувати стилі і полегшує читання коду.
3. Міксіни: Міксини дозволяють визначати набір стилів, які можуть бути використані повторно в різних місцях. За допомогою міксинів можна створювати стандартизовані набори стилів і використовувати їх для елементів з однаковими властивостями.
4. Функції: Препроцесори дозволяють використовувати функції для виконання різних операцій над значеннями, таких як математичні обчислення, операції з кольорами тощо.
5. Імпорт: Можна імпортувати стилеві файли один в одного, що полегшує організацію та підтримку стилів.

Синтаксис LESS і SCSS трохи відрізняється, але вони пропонують подібні можливості. SCSS є розширеним синтаксисом SASS, що включає функціональність LESS та зберігає звичний синтаксис CSS, що робить його більш сумісним з наявними стилями у проектах.

Для пришвидшення процесу створення стилів використовують CSS фреймворки.

CSS-фреймворки - це набір готових CSS-стилів, компонентів та рішень, які допомагають розробникам швидко та ефективно стилізувати веб-сторінки та створювати привабливі користувацькі інтерфейси. Вони забезпечують готовий набір стилів, шаблонів та компонентів, що дозволяє зосередитись на функціональності та розробці, не витрачаючи багато часу на написання CSS-стилів з нуля.

Ось декілька популярних CSS-фреймворків:

Bootstrap: Bootstrap - це один з найпопулярніших CSS-фреймворків, розроблений командою Twitter. Він надає готові компоненти, сітку, стилі кнопок, форм, навігації, таблиць та інших елементів інтерфейсу. Bootstrap також має вбудовану підтримку адаптивного дизайну, що дозволяє просто створювати веб-сторінки, що працюють на різних пристроях та екранах.

Foundation: Foundation - це інший популярний CSS-фреймворк, який також пропонує набір компонентів, сітку, стилі кнопок, форм, навігації та багато іншого. Foundation має потужну систему сітки та можливості розширення, що дозволяють створювати різноманітні дизайни та пристосовувати фреймворк до потреб проекту.

Bulma: Bulma - це легкий та модульний CSS-фреймворк, який надає гнучкий набір класів для стилізації веб-сторінок. Він пропонує компоненти, сітку, навігацію, форми, кнопки та інші елементи інтерфейсу. Bulma володіє простим та чистим дизайном, а також підтримує адаптивний дизайн.

Tailwind CSS: Tailwind CSS - це інший модульний CSS-фреймворк, який пропонує набір класів для створення стилів. Він використовує концепцію "utility-first", де кожен клас має свою специфічну стилістику. Tailwind CSS дозволяє швидко та гнучко створювати власні стилі, використовуючи комбінації класів.

Material-UI: Material-UI - це CSS-фреймворк, який реалізує принципи дизайну Material Design від Google. Він надає готові компоненти, такі як кнопки, карточки, форми, навігація та багато іншого. Material-UI має елегантний та сучасний дизайн, що допомагає створювати стильні користувацькі інтерфейси.

Ці фреймворки дозволяють значно спростити процес розробки та забезпечують багатий набір готових стилів та компонентів для швидкої розробки веб-інтерфейсів. Вибір CSS-фреймворку залежить від ваших вимог, дизайну та уподобань розробки.

Зауважимо, що одним із найбільш зручним для освоєння початківцями є фреймворк Bootstrap.

Bootstrap є одним з найпопулярніших CSS-фреймворків, розроблених командою Twitter. Він надає готові стилі, компоненти та шаблони, що допомагають швидко розробляти привабливі та респонсивні веб-сторінки. Основні особливості та можливості Bootstrap включають наступне:

- Сітка: Bootstrap має потужну систему сітки, що дозволяє легко створювати розташування елементів на сторінці. Сітка побудована на основі колонок і рядків, де сторінку можна розділити на 12 колонок. Він надає класи, що дозволяють розміщувати елементи в різних комбінаціях колонок та рядків для створення гнучкого та адаптивного макету.
- Готові компоненти: Bootstrap надає велику кількість готових компонентів, таких як кнопки, форми, картки, навігація, панелі, модальні вікна, слайдери, переключателі, списки та багато іншого. Ці компоненти мають готові стилі та інтерактивні функції, що дозволяють швидко додавати функціональність до веб-сторінок.
- Теми та налаштування: Bootstrap дозволяє настроювати тему та стиль фреймворку за допомогою Sass-змінних або використовувати вже готові теми. Це дає можливість відрізняти веб-сторінки за вашим бажанням та дизайном проекту.
- Адаптивний дизайн: Bootstrap підтримує адаптивний дизайн, що дозволяє створювати веб-сторінки, що працюють на різних пристроях та екранах. Він має вбудовану підтримку медіа-запитів та класів, що дозволяють показувати, приховувати або змінювати вигляд елементів в залежності від розміру екрану.
- Документація та спільнота: Bootstrap має детальну документацію, яка надає пояснення та приклади використання всіх компонентів та можливостей фреймворку. Також, він має велику спільноту розробників, де можна знайти підтримку, шаблони, розширення та інші корисні ресурси.

Bootstrap є дуже популярним фреймворком, оскільки він дозволяє розробникам швидко створювати стильні та привабливі веб-сторінки з мінімальними зусиллями. Він є гнучким, масштабованим та підходить для різних типів проектів.

1.2.3. МОВА JAVASCRIPT ТА ЇЇ ОСОБЛИВОСТІ

JavaScript є мовою програмування, яка використовується у веб-розробці для створення динамічних інтерактивних елементів на веб-сторінках. Вона працює на боці клієнта (у веб-браузері) і надає можливості взаємодії з користувачем, маніпулювання веб-елементами, обробки подій та виконання різних операцій.

Основні аспекти та функції JavaScript у веб-розробці включають:

Взаємодія з користувачем: JavaScript дозволяє створювати динамічні ефекти та інтерактивність на веб-сторінках. Це можуть бути валідація форм, анімації, взаємодія з кнопками, перетягування елементів, відображення спливаючих вікон та багато іншого.

Маніпуляція з DOM: JavaScript надає доступ до Document Object Model (DOM), який представляє структуру веб-сторінки. За допомогою JavaScript можна маніпулювати елементами DOM, змінювати їх властивості, стилі, додавати та видаляти елементи, змінювати контент сторінки динамічно.

Обробка подій: JavaScript дозволяє обробляти події, такі як клік, наведення курсора, ввод даних, зміна розміру вікна браузера та багато інших. Це дозволяє реагувати на дії користувача і виконувати певні дії при виникненні певних подій.

Взаємодія з сервером: JavaScript може взаємодіяти з сервером, виконуючи запити Ajax (асинхронні JavaScript та XML). Це дозволяє отримувати та відправляти дані на сервер без перезавантаження сторінки, що забезпечує плавну та динамічну взаємодію з сервером.

Розробка веб-додатків: JavaScript є ключовою мовою для розробки веб-додатків. З його допомогою можна створювати повноцінні клієнтські додатки, які працюють у браузері та надають багатофункціональність, таку як відправка даних на сервер, аутентифікація, обробка даних та багато іншого.

JavaScript є однією з основних мов програмування у веб-розробці, і вона використовується для надання багатого досвіду користувача, взаємодії зі сторінками та створення складних веб-додатків. Синтаксис JavaScript включає набір правил і конструкцій, які використовуються для написання програм на цій мові. Основні елементи синтаксису JavaScript включають наступне:

Змінні: Для оголошення змінних використовується ключове слово `var`, `let` або `const`, за яким слідує ім'я змінної. Наприклад: `var x = 5;`.

Типи даних: JavaScript має декілька вбудованих типів даних, включаючи числа, рядки, булеві значення, об'єкти, масиви та інші.

Оператори: JavaScript підтримує різні оператори, такі як арифметичні (`+`, `-`, `*`, `/`), порівняння (`<`, `>`, `<=`, `>=`, `==`, `===`, `!=`, `!==`), логічні (`&&`, `||`, `!`) та інші.

Умовні конструкції: Для виконання умовної логіки використовуються конструкції `if`, `else if` та `else`. Наприклад:

```
if (умова) {
  // Виконується, якщо умова істинна
} else if (інша умова) {
  // Виконується, якщо інша умова істинна
} else {
  // Виконується, якщо жодна з умов не істинна
}
```

Цикли: Для повторення фрагментів коду використовуються цикли, такі як `for`, `while`, `do while`. Наприклад:

```
for (ініціалізація; умова; ітерація) {
  // Виконується, доки умова істинна
}

while (умова) {
  // Виконується, доки умова істинна
}

do {
  // Виконується хоча б один раз, після чого перевіряється умова
} while (умова);
```

Функції: JavaScript дозволяє оголошувати функції, що містять блок коду, який можна викликати у різних частинах програми. Наприклад:

```
function назва_функції(параметри) {
```

```
  // Код функції
}

// Виклик функції
назва_функції(аргументи);
```

Обробка подій: JavaScript може реагувати на події, такі як клік миші, натискання клавіші, наведення курсора тощо, за допомогою обробників подій. Наприклад:

```
елемент.addEventListener('click', function() {
  // Код, що виконується при кліку на елемент
});
```

Це лише кілька основних елементів синтаксису JavaScript. Мова має багато інших можливостей та конструкцій, які дозволяють розробникам створювати складні програми та динамічні веб-додатки.

JavaScript (JS) є мовою програмування, в якій існує набір стандартів, які визначають її функціональність та можливості. Один з найпопулярніших стандартів - ECMAScript (ES). Найновіша версія стандарту ES називається ES6, або ECMAScript 2015. Ось деякі важливі аспекти стандартів мови JavaScript:

ECMAScript (ES): ECMAScript є стандартом, на основі якого побудована мова JavaScript. Він визначає синтаксис, типи даних, об'єкти, функції та інші елементи мови. Версії ES охоплюють певний набір нововведень, які розширюють функціональність мови та поліпшують її роботу.

ES6 (ECMAScript 2015): ES6 є значним оновленням мови JavaScript, яке було випущено в 2015 році. Воно включає багато нових функцій та поліпшень, які полегшують розробку великих проектів та покращують читабельність коду. Деякі з основних нововведень ES6 включають:

- Константи (`const`) та блочні змінні (`let`) для оголошення змінних з обмеженими областями видимості.
- Стрілкові функції (`=>`) для коротшого синтаксису функцій.

- Розпорядник розширень об'єкта (`...`) для роботи зі списками аргументів та масивами.
- Класи для оголошення об'єктно-орієнтованих класів.
- Розширені можливості роботи з рядками, масивами, об'єктами та іншими типами даних.
- Проміси (`Promise`) для асинхронного програмування.

Підтримка браузерами: Різні версії стандарту ES підтримуються різними веб-браузерами. Найновіші версії браузерів підтримують більшість функцій ES6, але старіші версії можуть мати обмежену підтримку. Це може вимагати використання поліфілів або транспіляції коду для забезпечення сумісності з різними браузерами.

Послідовність стандартів: Після ES6 були випущені наступні версії стандарту ES, такі як ES7, ES8, ES9 та інші. Кожна нова версія включає додаткові функції та покращення, які доповнюють функціональність мови JavaScript.

У розробці веб-додатків важливо використовувати сучасні версії стандартів ES та знати їх можливості. Це дозволяє писати більш ефективний, читабельний та масштабований код. Деякі браузери та середовища розробки також підтримують функції ES, які ще не були офіційно включені до стандарту.

Є кілька способів підключення JavaScript до веб-сторінки. Ось декілька основних методів:

Вкладений скрипт: Найпростіший спосіб - використовувати тег `<script>` без атрибуту `src`. Ви можете вбудувати JavaScript-код безпосередньо всередині тегу `<script>`. Наприклад:

```
<script>
  // Ваш JavaScript-код
</script>
```

Зовнішній файл: Ще один популярний спосіб - підключити зовнішній файл JavaScript за допомогою атрибуту `src` тегу `<script>`. Ви можете створити окремий файл з розширенням `.js` та вказати його шлях у веб-сторінці. Наприклад:

```
<script src="script.js"></script>
```

Асинхронне завантаження: Якщо ви бажаєте завантажити JavaScript асинхронно, щоб не блокувати завантаження сторінки, ви можете використовувати атрибут `async`. Це підказує браузеру продовжувати завантаження сторінки, не чекаючи завершення завантаження JavaScript. Наприклад:

<script src="script.js" async></script>

Відкладене завантаження: Альтернативою асинхронному завантаженню є відкладене завантаження, яке можна досягти за допомогою атрибуту `defer`. Це підказує браузеру завантажити JavaScript, але виконати його після завершення завантаження сторінки. Наприклад:

<script src="script.js" defer></script>

Ви можете використовувати будь-яку з цих методів залежно від потреб вашого проекту. Зазвичай зовнішній файл JavaScript рекомендується для більших проектів, оскільки це полегшує організацію та керування кодом.

Для оптимізації та спрощення роботи з кодом використовуються спеціальні інструменти: Gulp та webpack, що є популярними інструментами для автоматизації рутинних завдань та збирання JavaScript-проектів. Вони дозволяють спростити процес розробки, оптимізувати код та забезпечити ефективну роботу з JavaScript-файлами.

Gulp: Gulp є інструментом для автоматизації рутинних завдань. Він працює на основі потоків (streams) та дозволяє виконувати різні завдання, такі як компіляція препроцесорів CSS або JavaScript, оптимізація зображень, об'єднання та мінімізація файлів тощо. Gulp використовує конфігураційний файл `gulpfile.js`, в якому вказуються завдання та їх налаштування.

Webpack: Webpack є інструментом для збирання та управління модулями в JavaScript-проектах. Він дозволяє розбити код на модулі, що полегшує розробку та підтримку великих проектів. Webpack може автоматично виявляти залежності між модулями та створювати оптимізовані пакети (bundles) зі зведеними JavaScript-файлами. Крім того, він підтримує використання додаткових інструментів, таких як лінтери (linter), компілятори препроцесорів CSS або Babel для роботи з сучасними функціями JavaScript.

Якщо говорити про їх використання разом, то Gulp та webpack можуть доповнювати один одного в проектах JavaScript. Gulp може використовуватись для запуску задач, таких як компіляція Sass, автоматичне перезавантаження сторінки, побудова розташування файлів тощо, тоді як webpack відповідає за збирання та оптимізацію JavaScript-коду, управління модулями та генерацію пакетів з кодом.

Загалом, Gulp та webpack допомагають забезпечити більш організований та ефективний процес розробки JavaScript-проектів, зменшуючи рутинні завдання та автоматизуючи процес збирання та оптимізації коду.

В “чистому” вигляді JS використовується не досить часто через великі затрати часу та ресурсів на розробку прикладного рішення. Тому, найбільш оптимальним є використання фреймворків.

JavaScript-фреймворки - це набір інструментів, бібліотек та готових рішень, які допомагають розробникам створювати веб-додатки та веб-інтерфейси швидко та ефективно. Фреймворки надають структуру, готові компоненти, розширюють функціональність JavaScript та спрощують процес розробки.

Ось декілька популярних JavaScript-фреймворків:

- React: React - це популярний фреймворк для розробки користувацьких інтерфейсів. Він використовує підхід на основі компонентів, що дозволяє розбити веб-сторінку на незалежні, перевикористовувані компоненти. React працює з віртуальним DOM та надає швидку та ефективну оновлення інтерфейсу при зміні даних. Він є частиною платформи React, що включає також React Native для розробки мобільних додатків.
- Angular: Angular - це повноцінний фреймворк для створення веб-додатків. Він надає широкий набір інструментів для розробки, таких як двустороннє зв'язування даних, маршрутизація, керування формами, розширюваність та тестування. Angular використовує TypeScript, розширення JavaScript, що надає статичну типізацію та інші функції для поліпшення розробки.
- Vue.js: Vue.js - це прогресивний фреймворк для розробки користувацьких інтерфейсів. Він надає простоту використання та інтуїтивний синтаксис. Vue.js має широкі можливості, такі як директиви, віртуальний DOM, розширення, компоненти та система роутингу. Він може використовуватись як в простих проектах, так і в складних веб-додатках.

- Ember.js: Ember.js - це фреймворк, який спрощує створення амбітних веб-додатків. Він надає структуру та конвенції, що полегшують розробку, включаючи систему шаблонів, маршрутизацію, керування станом, тестування та багато іншого. Ember.js підтримує підхід "батарейки включені", що означає, що він має вбудовані багато рішень із коробки.
- Node.js: Node.js - це платформа, побудована на движку JavaScript V8, що дозволяє виконувати JavaScript на серверному боці. Вона надає розширені можливості розробки серверних додатків, включаючи роботу з мережевими запитами, файловою системою, базами даних та іншими складовими серверної розробки.

Кожен з цих фреймворків має свої переваги та використовується для різних типів проектів та сценаріїв. Вибір фреймворку залежить від ваших потреб та вимог проекту.

1.2.4. МОВА PHP ТА ЇЇ ОСОБЛИВОСТІ

PHP (Hypertext Preprocessor) є мовою програмування, яка широко використовується для розробки веб-додатків та динамічних веб-сторінок. Ось деякі особливості та використання мови PHP:

Скриптова мова: PHP є скриптовою мовою програмування, що означає, що вона виконується на стороні сервера. Код PHP обробляється на веб-сервері, а результати відправляються до клієнта у вигляді статичної HTML-сторінки. Це дозволяє створювати динамічні веб-сторінки, які можуть взаємодіяти з базами даних, обробляти форми, генерувати контент на льоту та багато іншого.

Веб-розробка: PHP є одним з найпопулярніших виборів для веб-розробки. Він надає велику кількість функцій та інструментів для створення веб-додатків, включаючи роботу з базами даних, обробку форм, сеанси, роботу з файлами, роботу з API та багато іншого. PHP підтримує різні протоколи та стандарти веб-розробки, такі як HTTP, XML, JSON та інші.

Робота з базами даних: PHP має вбудовану підтримку для роботи з різними типами баз даних, включаючи MySQL, PostgreSQL, SQLite та інші. Він дозволяє взаємодіяти з базами даних, виконувати запити, отримувати, вставляти, оновлювати та видаляти дані. Це робить PHP потужним інструментом для створення веб-додатків, що взаємодіють з базами даних.

Розширюваність: PHP має велику спільноту розробників та багато розширень, бібліотек та фреймворків, що допомагають прискорити процес розробки та розширити функціональність. Наприклад, фреймворки Laravel, Symfony, CodeIgniter та Yii дозволяють розробляти веб-додатки за певними шаблонами та надають готові рішення для роботи з маршрутизацією, шаблонами, базами даних та іншими аспектами розробки.

Велика спільнота: PHP має велику та активну спільноту розробників, яка надає підтримку, документацію, онлайн-курси та багато інших ресурсів. Це робить PHP доступним для навчання та дозволяє швидко знайти відповіді на питання або допомогу при розробці.

Загалом, PHP використовується для розробки динамічних веб-сторінок та веб-додатків. Він надає широкі можливості для роботи з базами даних, обробки форм, генерації контенту та інших завдань, що зазвичай вимагають серверної обробки.

Мова PHP має кілька версій, які випускаються регулярно з покращеннями та новими функціями. Ось кілька важливих версій PHP:

PHP 5: PHP 5 була важливою версією, що ввела багато нових функцій та покращень. Вона включала підтримку об'єктно-орієнтованого програмування, магічних методів, просторів імен, винятків, функціоналу SPL (Standard PHP Library) та іншого.

PHP 7: PHP 7 була значною версією, що принесла покращену продуктивність та ефективність. Вона мала вдвічі швидший виконавчий двигун за рахунок переходу на Zend Engine 3.0. PHP 7 також включала нові можливості, такі як типізація скалярних аргументів та повернення, обробка помилок, оптимізації пам'яті та багато іншого.

PHP 8: PHP 8 була останньою стабільною версією мови PHP на момент мого навчання. Вона включала ще більше покращень продуктивності, новий синтаксис, покращену підтримку типізації, функції з покращеною роботою з рядками та масивами, нові функції ООП (такі як роздільне наслідування), динамічні властивості об'єктів та інші зміни.

Кожна нова версія PHP включає багато нових можливостей та покращень, що поліпшують продуктивність, безпеку, якість коду та розширюють функціонал мови. Добре триматися останньої стабільної версії PHP, оскільки вона має найбільшу підтримку та доступ до оновлень та нових функцій.

Синтаксис PHP базується на C-подібних мовах програмування, але має свої особливості та конструкції. Ось декілька ключових елементів синтаксису PHP:

Теги PHP: PHP-код включається в HTML-сторінку або в окремий PHP-файл за допомогою тегів `<?php` та `?>`. Наприклад:

```
<?php
// PHP-код тут
?>
```

Змінні: Змінні в PHP починаються з символу долара `$` та можуть містити букви, цифри та символ підкреслення. PHP є слабко типізованою мовою, тому не потрібно вказувати тип змінної. Наприклад:

```
$name = "John";
$age = 25;
```

Виведення: Для виведення значень або змінних на екран використовується функція `echo`. Наприклад:

```
echo "Hello, World!";
```

Умовні оператори: Умовні оператори, такі як `if`, `else`, `elseif`, дозволяють виконувати блоки коду на основі умов. Наприклад:

```
if ($age >= 18) {
    echo "You are an adult.";
} else {
    echo "You are a minor.";
}
```

Цикли: Цикли `for`, `while`, `do-while` використовуються для повторення блоків коду. Наприклад:

```
for ($i = 1; $i <= 5; $i++) {
    echo $i;
}
```

Масиви: Масиви в PHP можуть бути індексованими або асоціативними. Індексовані масиви мають числові індекси, а асоціативні - ключі-значення. Наприклад:

```
$numbers = array(1, 2, 3, 4, 5);
$person = array("name" => "John", "age" => 25);
```

Функції: Функції в PHP дозволяють групувати блоки коду для виконання певних завдань. Функції можуть мати параметри та повертати значення. Наприклад:

```
function greet($name) {
    echo "Hello, $name!";
}

greet("John");
```

Це лише декілька основних елементів синтаксису PHP. PHP має багато інших конструкцій та функцій, які дозволяють розробникам створювати складні та потужні веб-додатки. Документація PHP містить повну інформацію про синтаксис та можливості мови.

PHP 8 внесла кілька основних змін у синтаксис та можливості мови. Ось кілька особливостей синтаксису PHP 8:

Запуск настроєних виключень (Throw Expression): PHP 8 дозволяє використовувати виключення безпосередньо в виразах. Замість використання блоку `try-catch`, можна використовувати новий синтаксис `throw` виразу. Наприклад:

```
$value = $data['key'] ?? throw new Exception('Key not found');
```

Null-safe оператор (Nullsafe Operator): Для полегшення роботи з потенційно нульовими значеннями, PHP 8 представив null-safe оператор `?->`, який дозволяє надійно доступатись до властивостей та методів об'єктів, навіть якщо об'єкт є `null`. Наприклад:

```
$length = $object->getProperty()?->getLength();
```

Явне задання типу повернення (Return Type Declaration): PHP 8 дозволяє явно вказувати тип повернення для функцій та методів. Це допомагає забезпечити більшу безпеку та ясність коду. Наприклад:

```
function calculateSum(array $numbers): int {
    // код функції
    return $sum;
}
```

Поліморфізм над параметрами типу (Union Types): PHP 8 дозволяє вказувати поліморфність над параметрами типу, тобто можна вказувати, що параметр може бути одного з кількох типів. Наприклад:

```
function processValue(int|float $value) {
    // код функції
}
```

Заборона неявного приведення типів (Strict Typing): У PHP 8 можна ввімкнути режим строгого типу, що вимагає явного вказання типів для аргументів та повернення функцій. Це допомагає зменшити ризик помилок та неправильного використання типів даних.

Match вирази (Match Expression): В PHP 8 введено новий оператор `match`, який надає більш могутній та зручний спосіб розгалуження коду залежно від значень.

```
$result = match($value) {
    'A' => 'Option A',
    'B' => 'Option B',
    'C' => 'Option C',
    default => 'Default Option',
};
```

Це лише кілька особливостей синтаксису, які були впроваджені в PHP 8. Ця версія мови надала розробникам більшу гнучкість, безпеку та продуктивність при розробці веб-додатків.

Об'єктно-орієнтоване програмування (ООП) в PHP дозволяє створювати структуровані, модульні та повторно використовувані кодові блоки, називані класами та об'єктами. ООП засноване на концепції об'єктів, які є екземплярами класів, що містять дані (властивості) та функції (методи) для роботи з цими даними. Ось деякі ключові поняття та конструкції ООП в PHP:

Класи: Класи визначають структуру об'єктів та включають в себе властивості та методи. Вони визначають шаблон для створення об'єктів. Наприклад:

```
class Person {
    // Властивості
    public $name;
    public $age;

    // Методи
    public function greet() {
        echo "Hello, my name is " . $this->name;
    }
}
```

Об'єкти: Об'єкти є екземплярами класів. Їх створюють за допомогою ключового слова `new`. Наприклад:

```
$person = new Person();
$person->name = "John";
$person->age = 25;
$person->greet();
```

Конструктори: Конструктори - це спеціальні методи, які викликаються при створенні нового об'єкта. Вони використовуються для ініціалізації властивостей об'єкта. Наприклад:

```
class Person {
    public $name;
    public $age;

    public function __construct($name, $age) {
        $this->name = $name;
        $this->age = $age;
    }
}

$person = new Person("John", 25);
```

Наслідування: Наслідування дозволяє створювати класи, які успадковують властивості та методи від батьківського класу. Воно допомагає створювати ієрархію класів та реалізувати концепцію поліморфізму. Наприклад:

```
class Student extends Person {
    public $studentId;

    public function study() {
        echo $this->name . " is studying.";
    }
}

$student = new Student("Jane", 20);
$student->study();
```

Інкапсуляція: Інкапсуляція в PHP дозволяє обмежувати доступ до властивостей та методів класу. Зазвичай використовуються модифікатори доступу, такі як `public`, `private` та `protected`. Наприклад:

```
class Person {
    private $name;
    public $age;

    public function getName() {
        return $this->name;
    }

    public function setName($name) {
        $this->name = $name;
    }
}

$person = new Person();
$person->setName("John");
echo $person->getName();
```

Це лише декілька основних конструкцій об'єктно-орієнтованого програмування в PHP. Використання ООП дозволяє створювати більш структурований, гнучкий та легко утримуваний код для веб-розробки.

Мова PHP має свої переваги та недоліки, які варто враховувати при виборі її для веб-розробки. Ось кілька переваг та недоліків мови PHP:

Переваги мови PHP:

Широке використання: PHP є однією з найпопулярніших мов програмування для веб-розробки. Вона має велику спільноту розробників та велику кількість готових рішень, бібліотек, фреймворків та розширень, що спрощує розробку та розширення веб-додатків.

Легкість вивчення: PHP має простий та зрозумілий синтаксис, що робить його доступним для початківців. Велика кількість ресурсів, документації та прикладів допомагають швидко освоїти мову та почати розробку веб-додатків.

Велика підтримка баз даних: PHP має вбудовану підтримку для багатьох типів баз даних, таких як MySQL, PostgreSQL, SQLite та багато інших. Це робить PHP популярним вибором для розробки додатків, що взаємодіють з базами даних.

Швидкодія: З кожним випуском PHP вдосконалюється продуктивність та ефективність виконання коду. Останні версії PHP мають вдвічі швидший виконавчий двигун, що робить його швидким та підходящим для обробки великого обсягу запитів.

Недоліки мови PHP:

Слабка типізація: PHP є слабко типізованою мовою, що означає, що типи даних не завжди строго контролюються. Це може призводити до непередбачуваної поведінки та помилок, особливо у великих проектах.

Вразливості безпеки: Історично PHP мав деякі проблеми з безпекою, і некоректно налаштований веб-додаток може бути вразливим до атак, таких як внедрення кодування та перехоплення сесій.

Неоднорідність функцій та бібліотек: У попередніх версіях PHP була деяка неоднорідність у синтаксисі та поведінці функцій та бібліотек. Хоча PHP 7 і PHP 8 вдосконалили це, все ще можуть бути деякі розбіжності, особливо коли використовуються старі бібліотеки або фреймворки.

Масштабованість: При роботі з великими проектами або високим навантаженням PHP може зіткнутися з обмеженнями масштабованості. Деякі великі компанії віддають перевагу іншим мовам, таким як Java або Node.js, для масштабованих систем.

1.2.5. МОВА ЗАПИТІВ РЕЛЯЦІЙНИХ БАЗ ДАНИХ SQL

SQL (Structured Query Language) - це стандартна мова програмування для керування та роботи з реляційними базами даних. Вона використовується для збереження, зміни, вибору та видалення даних у базах даних. Ось декілька ключових аспектів SQL:

Типи команд SQL: SQL включає різні типи команд, такі як:

- `SELECT`: Використовується для вибору даних з бази даних. Запит може містити умови, сортування, групування та інші параметри для точного вибору даних.
- `INSERT`: Використовується для вставки нових записів у таблицю бази даних.
- `UPDATE`: Використовується для оновлення існуючих записів у таблиці бази даних.
- `DELETE`: Використовується для видалення записів з таблиці бази даних.

- `CREATE`: Використовується для створення нових таблиць, баз даних або інших об'єктів бази даних.
-
- `ALTER`: Використовується для зміни структури таблиць або інших об'єктів бази даних.
- `DROP`: Використовується для видалення таблиць, баз даних або інших об'єктів бази даних.

Умови та фільтрація: SQL дозволяє застосовувати умови для фільтрації даних. Це дозволяє вибрати записи, які відповідають певним критеріям. Наприклад:

*SELECT * FROM users WHERE age > 18;*

Сортування та групування: SQL дозволяє сортувати дані за певними полями та групувати дані за певними категоріями. Наприклад:

*SELECT * FROM products ORDER BY price DESC;*
SELECT category, COUNT(*) FROM products GROUP BY category;

Об'єднання таблиць: SQL дозволяє об'єднувати дані з різних таблиць за певними умовами. SQL JOIN - це операція, яка дозволяє комбінувати дані з двох або більше таблиць на основі певного зв'язку між ними. Вона використовується для об'єднання рядків даних з різних таблиць на основі спільних значень стовпців.

У SQL є кілька типів JOIN, які визначають, які рядки даних будуть включені в результат. Ось декілька основних типів JOIN:

INNER JOIN: INNER JOIN повертає тільки ті рядки, для яких є спільні значення в обох таблицях. Він використовується для отримання спільних даних з двох або більше таблиць. Наприклад:

SELECT orders.order_id, customers.customer_name
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id;

LEFT JOIN: LEFT JOIN повертає всі рядки з першої (лівої) таблиці і відповідні рядки з другої (правої) таблиці. Якщо немає відповідних рядків у другій таблиці, то для них будуть встановлені значення NULL. Наприклад:

```
SELECT customers.customer_name, orders.order_id
FROM customers
LEFT JOIN orders ON customers.customer_id = orders.customer_id;
```

RIGHT JOIN: RIGHT JOIN повертає всі рядки з другої (правої) таблиці і відповідні рядки з першої (лівої) таблиці. Якщо немає відповідних рядків у першій таблиці, то для них будуть встановлені значення NULL. Наприклад:

```
SELECT customers.customer_name, orders.order_id
FROM customers
RIGHT JOIN orders ON customers.customer_id = orders.customer_id;
```

FULL JOIN (або FULL OUTER JOIN): FULL JOIN повертає всі рядки з обох таблиць, незалежно від того, чи є вони спільними. Якщо немає відповідних рядків у першій або другій таблиці, то для них будуть встановлені значення NULL. Проте FULL JOIN не підтримується в усіх реляційних базах даних.

```
SELECT customers.customer_name, orders.order_id
FROM customers
FULL JOIN orders ON customers.customer_id = orders.customer_id;
```

Це лише опис основних типів JOIN в SQL. JOIN.

Практичного використання мова SQL отримує через системи керування базами даних. Розглянемо дві з них.

MySQL і MariaDB є двома популярними системами управління базами даних (СУБД), які базуються на тій же технології та використовують SQL для роботи з даними. Однак, є кілька відмінностей між ними. Ось детальніше про кожну з цих систем.

Спочатку про MySQL:

- MySQL була розроблена у 1994 році та випущена компанією MySQL AB. Вона стала однією з найпопулярніших СУБД у світі та є відкритою програмною системою з вільною ліцензією GPL.
- MySQL підтримує широкий спектр функцій та може використовуватись для розробки різноманітних веб-додатків, включаючи великі веб-портали та системи управління контентом.
- MySQL має розгортання як на одному сервері, так і в розподіленій архітектурі з використанням реплікації та кластерів.
- Ключові риси MySQL включають швидкодію, надійність, простоту в установці та налаштуванні, а також велику кількість доступних драйверів для різних мов програмування.

MariaDB:

- MariaDB є форком MySQL, розробленим після придбання компанією Sun Microsystems, що володіла MySQL, компанією Oracle Corporation. MariaDB створена з метою збереження відкритості та розвитку СУБД.
- MariaDB зберігає високу сумісність з MySQL, що означає, що більшість програм, розроблених для MySQL, без змін працюють на MariaDB.
- MariaDB пропонує деякі додаткові функції та покращення в порівнянні з MySQL, включаючи підтримку нових типів даних, швидшу роботу, покращену оптимізацію запитів та більшу безпеку.
- Велика спільнота розробників підтримує Maria DB, надаючи оновлення, патчі та допомогу з вирішення проблем.

Основна відмінність між MySQL та MariaDB полягає в тому, що MariaDB є форком MySQL і надає певні покращення, але залишається сумісною з MySQL. У випадку вибору між ними, рішення залежить від конкретних потреб проекту, але обидві системи є потужними та популярними варіантами для управління базами даних.

Архітектура баз даних для SQL (Structured Query Language) включає декілька ключових компонентів, включаючи таблиці, структури даних та зв'язки між ними. Основні поняття в архітектурі баз даних SQL включають:

Таблиці: Таблиці є основними компонентами баз даних SQL. Вони представляють сховище для зберігання даних і мають рядки (records) і стовпці (columns). Кожен стовпець представляє окреме поле даних, а кожен рядок представляє окремий запис зі значеннями для кожного поля. Кожна таблиця має унікальне ім'я та може мати первинний ключ (primary key), який ідентифікує унікальні записи в таблиці.

Структури даних: Для зберігання даних у таблицях використовуються різні типи структур даних, такі як рядки, числа, дати, булеві значення тощо. SQL надає різні типи даних для зберігання різних видів інформації, і розробники можуть визначати власні користувацькі типи даних.

Ключі: Ключі використовуються для ідентифікації та встановлення зв'язків між таблицями. Первинний ключ (primary key) є унікальним ідентифікатором для кожного запису в таблиці. Зовнішній ключ (foreign key) використовується для зв'язку з іншою таблицею та встановлення залежностей між даними.

Зв'язки між таблицями: В SQL використовуються різні типи зв'язків між таблицями для встановлення зв'язків та залежностей між даними. Це можуть бути один до одного (one-to-one), один до багатьох (one-to-many) та багато до багатьох (many-to-many) зв'язки. Зв'язки використовують зовнішні ключі та інші механізми для забезпечення цілісності даних.

Нормалізація даних є процесом організації даних у базі даних для забезпечення ефективності, цілісності та уникнення дублювання даних. Вона включає розбиття таблиць на менші, більш спеціалізовані таблиці та встановлення зв'язків між ними. Нормалізація даних допомагає покращити швидкодію, зменшити розмір бази даних та запобігти втраті даних.

Нормалізація даних складається з кількох рівнів (нормальних форм), таких як перша нормальна форма (1NF), друга нормальна форма (2NF), третя нормальна форма (3NF) та інші. Кожна нормальна форма має свої вимоги щодо організації даних та зв'язків між таблицями.

Загалом, архітектура баз даних SQL включає таблиці, структури даних, ключі та зв'язки, а нормалізація даних допомагає забезпечити ефективну та організовану структуру бази даних.

MySQL підтримує різноманітні типи даних, які використовуються для збереження різних видів інформації. Ось кілька основних типів даних, які використовуються в MySQL:

Числові типи даних:

- INT: Використовується для збереження цілих чисел. Наприклад, INT(10) зберігає ціле число до 10 цифр.

- FLOAT та DOUBLE: Використовуються для збереження чисел з плаваючою точкою. FLOAT використовує менше місця для збереження, ніж DOUBLE, але DOUBLE забезпечує більшу точність.
- DECIMAL: Використовується для точних обчислень з фіксованою точністю.

Рядкові типи даних:

- VARCHAR: Використовується для збереження рядків змінної довжини. Наприклад, VARCHAR(255) зберігає рядок до 255 символів.
- CHAR: Використовується для збереження рядків фіксованої довжини. Наприклад, CHAR(10) зберігає рядок завжди довжиною 10 символів.
- TEXT: Використовується для збереження довгих текстових значень.

Дата та часові типи даних:

- DATE: Використовується для збереження дати. Формат: 'YYYY-MM-DD'.
- TIME: Використовується для збереження часу. Формат: 'HH:MM:SS'.
- DATETIME: Використовується для збереження дати та часу. Формат: 'YYYY-MM-DD HH:MM:SS'.

Логічний тип даних:

- BOOLEAN або BOOL: Використовується для збереження значень true або false.

Бінарні типи даних:

- BLOB: Використовується для збереження бінарних даних, таких як зображення або великі файли.
- VARBINARY: Аналогічний VARCHAR, але для збереження бінарних даних змінної довжини.

Спеціалізовані типи даних:

- ENUM: Використовується для збереження одного значення з визначеного набору можливих значень.
- SET: Використовується для збереження декількох значень з визначеного набору можливих значень.

Це лише кілька основних типів даних, які підтримує MySQL. Існують інші типи даних, такі як JSON, TIMESTAMP, YEAR, і багато інших, які можуть бути використані залежно від потреб вашого проекту.

Для зручного управління базами даних використовують спеціалізовані програмні рішення - менеджери баз даних. Розглянемо один з них.

phpMyAdmin є безкоштовним і відкритим інструментом для адміністрування баз даних MySQL через веб-інтерфейс. Він надає зручний спосіб керування базами даних без необхідності використовувати командний рядок або писати власний код.

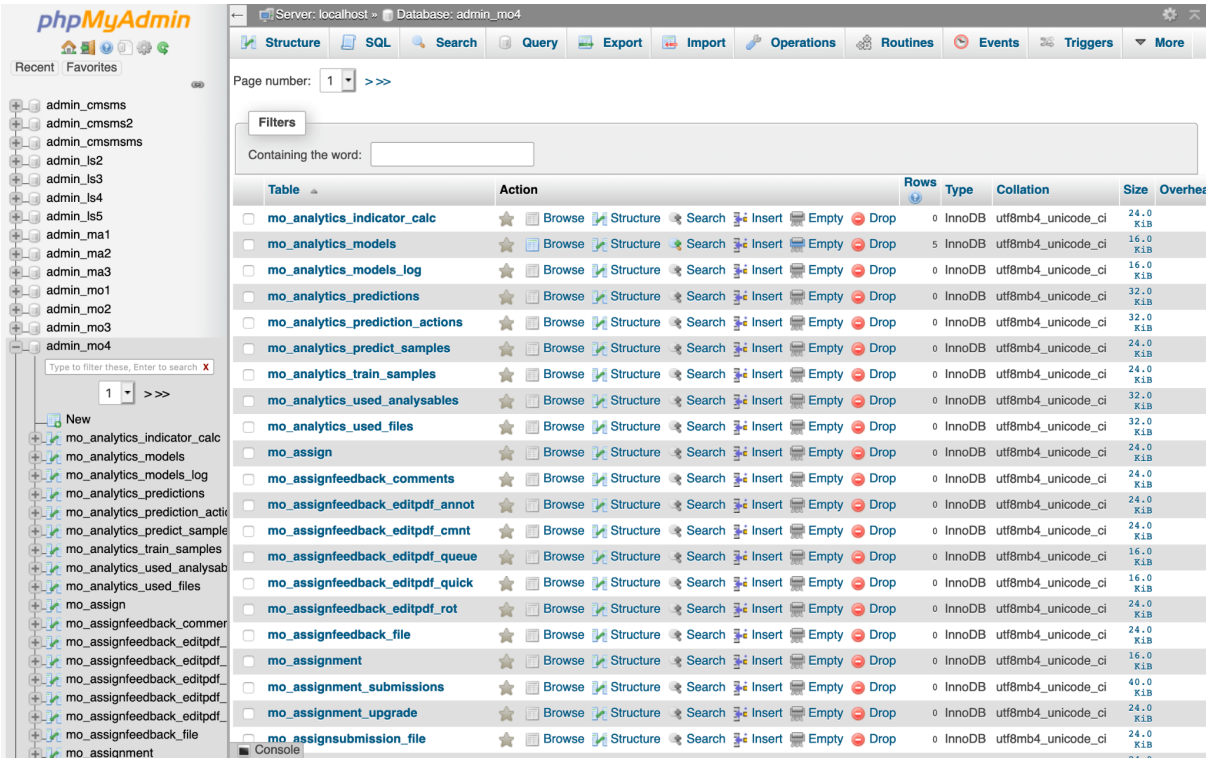

Рис. 2. Інтерфейс PHPMyAdmin

Ось декілька ключових рис і функцій phpMyAdmin:

Управління базами даних: phpMyAdmin дозволяє створювати, видаляти та модифікувати бази даних. Ви можете створювати нові таблиці, редагувати їх структуру, видаляти та переміщувати дані.

Виконання запитів SQL: phpMyAdmin надає можливість виконувати запити SQL безпосередньо з веб-інтерфейсу. Ви можете виконувати SELECT-запити для вибору даних, а також виконувати INSERT, UPDATE та DELETE-запити для зміни даних у базі даних.

Управління користувачами: phpMyAdmin дозволяє створювати нових користувачів, надавати їм права доступу до баз даних та змінювати їх налаштування. Ви можете керувати рівнями доступу, паролями та привілеями користувачів.

Імпорт та експорт даних: phpMyAdmin надає можливість імпортувати та експортувати дані в різних форматах, таких як SQL, CSV або XML. Ви можете імпортувати готові резервні копії баз даних або експортувати дані для збереження або обміну.

Перегляд та редагування даних: phpMyAdmin надає можливість переглядати та редагувати дані в таблицях бази даних. Ви можете додавати, видаляти та змінювати записи в таблицях, а також сортувати, фільтрувати та шукати дані.

Візуалізація структури бази даних: phpMyAdmin надає візуальне представлення структури бази даних у вигляді діаграми ER (Entity-Relationship). Це допомагає легше розуміти зв'язки між таблицями та їх структуру.

phpMyAdmin є потужним інструментом для роботи з базами даних MySQL через веб-інтерфейс. Він дозволяє легко і зручно керувати базами даних, виконувати запити SQL та здійснювати імпорт та експорт даних.

Крім безпосереднього менеджменту баз даних, в процесі роботи виникає необхідність відлагодження запитів.

Відлагодження запитів (query debugging) в MySQL відіграє важливу роль у виявленні та виправленні помилок в запитах до бази даних. Ось кілька способів відлагодження запитів в MySQL:

- Використання оператора `EXPLAIN`: Оператор `EXPLAIN` дозволяє отримати інформацію про те, як MySQL виконує запит та які індекси використовуються. Він допомагає виявити потенційні проблеми з продуктивністю, такі як повільні запити або використання неправильних індексів.
- Включення режиму відлагодження: Можна включити режим відлагодження (`--debug`, `--debug=d:t:i:o,/path/to/logfile`) під час виконання запитів або запуску MySQL сервера. Це створює відлагоджувальні дані, такі як інформацію про операції, виконані запити, розмір буферів тощо, які допомагають виявити проблеми та проаналізувати їх.
- Використання профілювання запитів: MySQL має можливість профілювання запитів, яка дозволяє відстежувати та аналізувати час виконання запитів, кількість рядків, оброблених запитом, використання ресурсів та інші метрики продуктивності. Це допомагає виявити повільні запити та покращити продуктивність.

- Використання інструментів відладки: Існують різні інструменти відладки для MySQL, які надають зручний інтерфейс для аналізу та відлагодження запитів. Наприклад, MySQL Workbench, Navicat, HeidiSQL тощо. Вони дозволяють виконувати запити, аналізувати результати, переглядати та аналізувати структуру бази даних, переглядати відлагоджувальні дані тощо.
- Використання логу запитів: MySQL може записувати логи запитів, які містять інформацію про виконані запити та помилки. Це дозволяє виявляти та аналізувати проблемні запити. Логи можуть бути ввімкнені шляхом налаштування конфігураційного файлу MySQL.

Ці методи дозволяють відлагоджувати запити в MySQL та виявляти та виправляти проблеми з продуктивністю або помилками. Залежно від конкретної ситуації може бути використано один або кілька з цих методів для ефективного відлагодження запитів.

Питання для самоконтролю:

1. Для чого призначена мова HTML?
2. За яким принципом працюють теги?
3. Що таке семантика мови HTML?
4. Для чого призначені селектори в CSS?
5. Яке призначення мають LESS / SASS?
6. В чому полягає особливість ES6?
7. Чим відрізняються змінні та константи в JS?
8. Як оголосити цикл з передумовою в JS?
9. Як оголошуються класи в PHP?
10. Які особливості надає PHP8?
11. Як будуються запити в SQL?
12. Для чого призначені оператори об’єднання (Joins)?

1.3. ЛОКАЛЬНЕ ТА СЕРВЕРНЕ ОТОЧЕННЯ

Для розробки на PHP зазвичай використовуються два типи оточень: локальне та серверне. Кожне з них має свої особливості та використовується для різних етапів розробки.

Локальне оточення (Local Environment):

Локальне оточення встановлюється безпосередньо на ваш комп'ютер і дозволяє вам розробляти та тестувати свої PHP-додатки на ньому, не підключаючись до живого сервера. Основні компоненти локального оточення включають:

- Веб-сервер: Наприклад, Apache, Nginx або Microsoft IIS. Він відповідає за обробку HTTP-запитів і виконання PHP-скриптів.
- Сервер бази даних: Наприклад, MySQL, MariaDB або PostgreSQL. Це дозволяє вам зберігати та отримувати доступ до даних в вашому локальному середовищі.
- PHP-інтерпретатор: Встановлюється PHP на ваш комп'ютер, що дозволяє виконувати PHP-код.

До переваг локального оточення відносяться:

- Зручність та швидкість розробки, оскільки ви можете працювати безпосередньо на своєму комп'ютері.
- Відсутність залежності від інтернет-з'єднання або доступу до сервера.

До недоліків локального оточення можуть відноситися:

- Відсутність реального сервера та його конфігурації, яка може впливати на певні аспекти виробничого середовища.
- Відмінності в конфігурації та версіях програмного забезпечення між локальним та серверним середовищами.

Серверне оточення (Server Environment):

Серверне оточення використовується для розгортання та запуску вашого PHP-додатку на реальному сервері, який доступний через Інтернет. Це може бути власний сервер або хостинг-провайдер. Основні компоненти серверного оточення включають:

- Веб-сервер: Наприклад, Apache, Nginx або Microsoft IIS.
- Сервер бази даних: Наприклад, MySQL, MariaDB або PostgreSQL.
- PHP-інтерпретатор: Встановлюється на сервері та виконує PHP-код.

Переваги серверного оточення включають:

- Максимальна сумісність з виробничим середовищем, оскільки ви працюєте на реальному сервері, який імітує умови продукції.
- Забезпечення доступу до додатка через Інтернет, що дозволяє перевірити функціональність та продуктивність в реальному середовищі.

Недоліки серверного оточення:

- Залежність від стабільного інтернет-з'єднання та доступу до сервера.
- Обмежені можливості з конфігурації сервера, які можуть бути доступні відповідно до політик хостинг-провайдера.

У веб-розробці на PHP використовуються обидва типи оточень, причому локальне оточення використовується для розробки, тестування та налагодження, а серверне оточення - для розгортання та запуску реального веб-додатку.

Загалом як для локального так і серверного оточення використовують два підходи:

- Нативне оточення.
- Оточення із засобами контейнеризації.

Розглянемо обидва з них.

Нативне оточення для розробки на PHP означає встановлення необхідних компонентів (веб-сервера, PHP, СУБД) безпосередньо на вашому комп'ютері. Це означає, що ви будете мати повний контроль над конфігурацією та налаштуванням цих компонентів для своїх потреб розробки.

Для нативного оточення на PHP ви можете встановити окремі компоненти, такі як:

- Веб-сервер: Наприклад, Apache, Nginx або Microsoft IIS. Ви можете встановити веб-сервер на своєму комп'ютері та налаштувати його для обробки PHP-скриптів.
- PHP-інтерпретатор: Встановіть PHP на свій комп'ютер. Ви можете вибрати необхідну версію PHP та налаштувати параметри за потребою.
- Сервер бази даних: Наприклад, MySQL, MariaDB або PostgreSQL. Встановіть СУБД на своєму комп'ютері та налаштуйте його для роботи з вашим PHP-додатком.

Нативне оточення надає повну гнучкість та контроль, але вимагає деякої експертизи в конфігуруванні та управлінні компонентами.

Докер (Docker) - це інструмент для контейнеризації програмного забезпечення, який дозволяє упаковувати та запускати додатки в контейнерах. Docker дозволяє вам

створювати віртуальне середовище, яке містить усі необхідні компоненти для роботи вашого PHP-додатку, такі як веб-сервер, PHP-інтерпретатор та СУБД. Ви можете використовувати готові образи контейнерів або створити свій власний образ, який містить усі необхідні компоненти та налаштування.

Використання Docker має такі переваги:

- Портативність: Ви можете створити Docker-образ, який містить усі необхідні компоненти та налаштування, і розгортати його на будь-якому середовищі, де є підтримка Docker.
- Ізоляція: Контейнери Docker ізолюють ваш додаток від інших додатків та середовища, що дозволяє уникнути конфліктів та забезпечити чистоту середовища.
- Швидкість розгортання: Docker дозволяє швидко створювати та розгортати контейнери, що прискорює процес розробки та тестування.

Використання Docker для розробки на PHP дозволяє створити одноразове середовище, яке легко відтворити та поділитися з іншими членами команди. Крім того, ви можете використовувати Docker для локального розробки або налаштувати контейнери для виробничого середовища.

Обираючи між нативним оточенням та Docker, рішення залежить від ваших потреб, рівня експертизи та вимог проекту. Сучасний технологічний тренд вказує на переважне використання контейнеризації.

1.3.1. СТЕКИ LAMP ТА WAMP

Стеки LAMP та WAMP є популярними конфігураціями для розробки веб-додатків на PHP. Обидва стеки складаються з набору компонентів, які працюють разом для створення та запуску PHP-додатків. Основні компоненти включають:

LAMP означає Linux, Apache, MySQL/MariaDB та PHP. Ось опис кожного компонента:

Linux: Операційна система Linux, яка використовується як базова платформа для створення стека LAMP. Вона забезпечує інфраструктуру та середовище виконання для інших компонентів.

Apache: Веб-сервер Apache, який відповідає за обробку запитів HTTP та надсилання відповідей до веб-браузерів. Він забезпечує розподілення веб-сторінок та виконання PHP-скриптів.

MySQL/MariaDB: Система керування базами даних (СКБД) MySQL або MariaDB, яка використовується для збереження та керування даними. Вона забезпечує можливість створення баз даних, таблиць та виконання запитів.

PHP: PHP-інтерпретатор, який виконує PHP-код і генерує динамічні веб-сторінки. Він дозволяє взаємодіяти з базою даних, обробляти форми та виконувати інші функції, необхідні для розробки веб-додатків.

WAMP означає Windows, Apache, MySQL/MariaDB та PHP. Це аналогічна конфігурація до LAMP, але замість операційної системи Linux використовується Windows.

Головна відмінність між LAMP і WAMP полягає в операційній системі, на якій вони працюють. LAMP використовує Linux, що є відкритою операційною системою, тоді як WAMP використовує Windows, яка є комерційною операційною системою від Microsoft.

Якщо ви працюєте на Linux або вам доступна віртуальна машина з Linux, LAMP може бути хорошим вибором, оскільки він надає розширені можливості конфігурування та керування. У разі, якщо ви працюєте на Windows, WAMP може бути більш зручним, оскільки він підтримує Windows як операційну систему.

В обох випадках, LAMP та WAMP, ви можете налаштувати й керувати веб-сервером, СКБД та PHP-інтерпретатором для розробки та тестування веб-додатків на PHP.

Варто зауважити, що існує також технологічний стек MAMP, побудований на базі операційної системи macOS. Він є досить поширеним серед професійних розробників, але через високу вартість платформи на базі Macbook або Mac mini - мало поширений серед початківців. Через це ми не будемо приділяти йому увагу в контексті даного підручника. Варто зауважити, що MAMP стек за практичним застосуванням дуже близький до LAMP.

Автори, покладаючись на тривалий практичний досвід, рекомендують використовувати для навчання саме LAMP стек. Використання LAMP (Linux, Apache, MySQL/MariaDB, PHP) для навчання веб-розробці має декілька переваг:

- Відкритий джерела: Всі компоненти LAMP є відкритими джерелами і безкоштовно доступні для використання. Це означає, що ви можете легко отримати доступ до програмного забезпечення та налаштувати їх для своїх потреб.
-

- Широке сприйняття: LAMP є однією з найпоширеніших конфігурацій для веб-розробки на PHP. Багато веб-хостинг-провайдерів та серверів підтримують LAMP, тому навички, набуті в навчанні з LAMP, можуть бути легко перенесені до реальних проектів.
- Гнучкість та контроль: Використання LAMP надає гнучкість і контроль над усіма компонентами стеку. Ви можете налаштувати сервер, базу даних та PHP-інтерпретатор відповідно до своїх потреб. Це дозволяє вам досліджувати різні налаштування та оптимізації для отримання найкращої продуктивності та безпеки.
- Розширюваність: Завдяки великій кількості розширень, бібліотек та фреймворків, доступних для PHP, ви можете побудувати різноманітні та потужні веб-додатки. LAMP надає основу для розробки великих проектів, а PHP є однією з найпоширеніших мов програмування для веб-розробки.
- Ресурсна доступність: Завдяки великій кількості документації, онлайн-ресурсів, форумів та спільнот, пов'язаних з LAMP та PHP, ви зможете знайти відповіді на свої питання, вирішити проблеми та швидко навчитися новим технологіям та практикам.

Загалом, LAMP є надійним, широко використовуваним та доступним стеком для навчання веб-розробці на PHP. Він дозволяє вам побудувати потужні веб-додатки, набути необхідних навичок та зрозуміти основні принципи веб-розробки.

1.3.2. ВЕБ-СЕРВЕРИ APACHE ТА NGINX

Apache і Nginx є двома з найпопулярніших веб-серверів, які використовуються для розгортання веб-додатків і обробки запитів HTTP. Обидва сервери мають свої особливості та застосування.

- Apache HTTP Server (зазвичай називається просто Apache) є одним з найстаріших і найбільш поширених веб-серверів. Деякі особливості Apache включають:
- Кросплатформенність: Apache підтримує різні операційні системи, такі як Linux, Windows, macOS тощо. Це робить його універсальним веб-сервером.
- Розширені можливості: Apache має широкий спектр модулів та розширень, що дозволяють розширити його функціональність.

Наприклад, можливості з перенаправлення URL, SSL-шифрування, автентифікації користувачів та багато іншого.

- Надійність та стабільність: Apache відомий своєю стабільністю та надійністю. Він здатний обробляти великі завантаження і забезпечувати високу продуктивність веб-додатків.

Nginx (вимовляється "Engine X") є веб-сервером та проксі-сервером, який набуває все більшої популярності в останні роки. Особливості Nginx включають:

- Висока продуктивність: Nginx був розроблений з урахуванням високої продуктивності і швидкості обробки запитів. Він ефективно використовує системні ресурси і може обробляти багато одночасних з'єднань.
- Низьке споживання ресурсів: Nginx відомий своєю низькою витратою пам'яті і процесорного часу, що дозволяє ефективно використовувати обмежені ресурси сервера.

Розширені можливості проксі-сервера: Nginx має вбудовані можливості проксі-сервера, які дозволяють балансувати навантаження на різні веб-сервери та керувати розподілом трафіку.

Висока масштабованість: Nginx може бути ефективно використаний для обробки великого обсягу запитів і розподілу навантаження на кластер серверів.

Обираючи між Apache і Nginx, варто враховувати специфічні потреби вашого проекту та особливості. Apache є більш універсальним і має широкий спектр функціональності, в той час як Nginx забезпечує високу продуктивність та низьке споживання ресурсів, особливо для сценаріїв з великою кількістю одночасних з'єднань.

Для встановлення Apache і Nginx на Ubuntu Linux виконайте наступні кроки:

Встановлення Apache:

- Відкрийте термінал на Ubuntu Linux.

- Виконайте наступну команду для встановлення Apache:

sudo apt update

sudo apt install apache2

Після встановлення Apache автоматично запуститься і почне слухати на порту 80.

Встановлення Nginx:

- Відкрийте термінал на Ubuntu Linux.
- Виконайте наступні команди для встановлення Nginx:

sudo apt update
sudo apt install nginx

Після встановлення Nginx автоматично запуститься і почне слухати на порту 80.

Налаштування фаєрволу:

Якщо на вашій системі використовується фаєрвол, такий як UFW, вам потрібно відкрити порти 80 для Apache або Nginx. Наприклад, для відкриття порту 80 для Apache виконайте команду:

sudo ufw allow 80

Перевірка роботи:

Після встановлення Apache або Nginx ви можете перевірити їх роботу, відкривши веб-браузер і введіть URL-адресу `http://localhost` або `http://<IP-адреса вашого сервера>`. Ви повинні бачити стандартну сторінку Apache або Nginx, відповідно.

Тепер ви встановили як Apache, так і Nginx на своєму Ubuntu Linux сервері. Ви можете використовувати їх для розгортання своїх веб-додатків або веб-сайтів. Пам'ятайте, що обидва веб-сервери не можуть працювати одночасно на одному порту, тому вам потрібно буде зупинити один сервер, щоб запустити інший.

При налаштуванні Nginx та Apache для локальної розробки наступні аспекти є важливими:

Порти: Переконайтеся, що веб-сервери використовують різні порти, оскільки вони не можуть працювати одночасно на одному порту. Зазвичай, Apache працює на порту 80, тоді як Nginx можна налаштувати на будь-який інший доступний порт, наприклад, 8080. Це дозволить вам запускати обидва сервери паралельно.

Кореневий каталог: Визначте кореневий каталог для вашого веб-сайту або веб-додатку. У випадку Apache це зазвичай каталог `/var/www/html`, а для Nginx - `/var/www`. Впевніться, що у вас є права доступу до цих каталогів для розробки і змін файлів.

Віртуальні хости: Використовуйте віртуальні хости для налаштування декількох сайтів або додатків на одному сервері. Це дозволить вам розділити різні проекти та використовувати різні доменні імена або піддомени для доступу до них. Для Apache це налаштовується через файли конфігурації в каталозі `/etc/apache2/sites-available`, а для Nginx - в каталозі `/etc/nginx/sites-available`.

Мови програмування та модулі: Переконайтеся, що необхідні мови програмування та модулі активовані у вашому веб-сервері. Наприклад, для PHP ви повинні мати встановлений PHP-інтерпретатор та відповідні модулі для Apache або Nginx.

Конфігурація SSL: Якщо ви плануєте використовувати HTTPS для шифрування комунікації, вам потрібно налаштувати SSL-сертифікат для вашого локального сервера. Для Apache це налаштовується через модуль `mod_ssl`, а для Nginx - за допомогою налаштування SSL-сертифікату в конфігураційному файлі.

Журнали: Переконайтеся, що ви налаштували журнали доступу та помилок, щоб відстежувати та аналізувати події, пов'язані з вашим сервером під час розробки.

Права доступу: Впевніться, що у вас є належні права доступу до відповідних файлів і каталогів, щоб ваш сервер міг читати і записувати файли, необхідні для розробки.

Загалом, при налаштуванні Nginx та Apache для локальної розробки варто звертати увагу на порти, кореневі каталоги, віртуальні хости, мови програмування, конфігурацію SSL, журнали та права доступу, щоб забезпечити належну роботу веб-серверів під час розробки вашого проекту.

Оновлення Nginx та Apache є важливим аспектом з точки зору безпеки, стабільності та функціональності веб-серверів. Ось деякі причини, чому важливо вчасно оновлювати ці сервери:

Безпека: Оновлення веб-серверів включають патчі та виправлення безпекових проблем, виявлених у попередніх версіях. Веб-сервери часто піддаються атакам, і оновлення допомагають усунути вразливості та запобігти потенційним вторгненням та компрометації сервера.

Виправлення помилок: Оновлення серверів Nginx та Apache також включають виправлення помилок та недоліків, які можуть вплинути на функціональність або працездатність веб-сервера. Вони можуть включати покращення продуктивності, підвищення стабільності та оптимізацію ресурсів.

Нові функції: Оновлення дозволяють отримати доступ до нових функціональних можливостей та покращень, які можуть поліпшити вашу розробку та досвід використання веб-серверів. Вони можуть включати підтримку нових протоколів, розширені налаштування, покращену підтримку PHP або інших мов програмування та багато іншого.

Сумісність: Оновлення можуть також включати покращення сумісності з новими версіями операційних систем, баз даних, браузерів та інших компонентів веб-серверного середовища. Це важливо для забезпечення сумісності та належного функціонування вашого веб-сервера в екосистемі веб-розробки.

Регулярне оновлення Nginx та Apache є ключовим елементом для забезпечення безпеки, надійності та продуктивності вашого веб-сервера. Важливо слідкувати за новими версіями, отримувати сповіщення про оновлення та вчасно виконувати процес оновлення, щоб захистити ваш сервер і забезпечити його ефективну роботу.

1.3.3. ІНТЕРПРЕТАТОР МОВИ PHP. ОСОБЛИВОСТІ ВЕРСІЙ PHP

Щоб встановити PHP на Ubuntu, виконайте наступні кроки:

Відкрийте термінал на Ubuntu Linux.

Оновіть список пакетів системи, виконавши наступну команду:

sudo apt update

Встановіть пакет `php` за допомогою наступної команди:

sudo apt install php

Ця команда встановить PHP зі стандартними пакетами та залежностями.

Після завершення встановлення перевірте, чи PHP встановлено правильно, виконавши команду:

php -v

Ця команда виведе вам версію PHP та іншу інформацію про встановлену версію.

Додаткові пакети PHP: Якщо вам потрібні деякі додаткові розширення PHP, ви можете встановити їх за допомогою команди `sudo apt install php-<назва розширення>`. Наприклад, якщо вам потрібне розширення MySQL, виконайте команду:

sudo apt install php-mysql

Перезапустіть веб-сервер: Після встановлення PHP може знадобитися перезапустити веб-сервер, щоб зміни набрали чинності. Для Apache виконайте команду:

sudo service apache2 restart

Для Nginx:

sudo service nginx restart

Це дозволить PHP взаємодіяти з веб-сервером.

Тепер ви встановили PHP на своєму Ubuntu-сервері. Ви можете перевірити його роботу, створивши PHP-скрипт і запустивши його у веб-браузері або перевіривши встановлені розширення PHP за допомогою функції `phpinfo()`.

PHP - це мова програмування, яка постійно розвивається та оновлюється. Особливості різних версій PHP можуть включати покращення продуктивності, нові функції, зміни в синтаксисі та інші вдосконалення. Ось кілька особливостей різних версій PHP:

PHP 5.x: PHP 5.x була широко використовуваною версією до появи PHP 7. Вона включає такі особливості, як підтримка об'єктно-орієнтованого програмування (ООП), розширення для роботи з базами даних, робота з XML та багато іншого.

PHP 7.x: PHP 7.x - це мажорне оновлення PHP, яке ввело численні покращення продуктивності та ефективності. Вона пропонує значні збільшення швидкості виконання, зменшення використання пам'яті, нові функції, включаючи типізацію аргументів та повернення, оператор null coalescing, вдосконалення генераторів та інші покращення.

PHP 8.x: PHP 8.x - остання версія PHP, що планується до релізу у 2020 році. Вона пропонує численні нові можливості та покращення. Деякі з особливостей PHP 8.x включають в себе: введення строго режиму типізації, спадкові типи, Union типи, атрибути, збільшену продуктивність, покращену обробку помилок та інші зміни.

Важливо відмітити, що при переході на нові версії PHP можуть виникати проблеми сумісності зі старим кодом, оскільки нові версії можуть вимагати використання нових синтаксичних конструкцій або мати змінений поведінку. Тому перед оновленням PHP варто ретельно перевірити сумісність вашого існуючого коду з новою версією та виконати відповідні зміни.

Крім того, кожна версія PHP має свою підтримку та життєвий цикл. Варто вибрати версію, яка надає необхідну функціональність та забезпечує активну підтримку та оновлення безпеки.

Налаштування PHP включає кілька кроків, які дозволяють змінити конфігурацію PHP для відповідності вашим потребам. Ось кілька основних кроків для налаштування PHP:

Знайдіть файл php.ini: Спочатку потрібно знайти файл php.ini, який містить конфігураційні параметри PHP. Зазвичай він розташовується в `/etc/php/<версія>/apache2/php.ini` для Apache або `/etc/php/<версія>/fpm/php.ini` для PHP-FPM.

Редагуйте файл php.ini: Відкрийте файл php.ini у текстовому редакторі і внесіть необхідні зміни. Основні параметри, які можна змінювати, включають:

- `memory_limit`: Встановлює обмеження пам'яті, доступної для PHP-сценаріїв.
- `max_execution_time`: Встановлює максимальний час виконання сценаріїв.
- `upload_max_filesize`: Встановлює максимальний розмір завантажуваних файлів.
- `post_max_size`: Встановлює максимальний розмір POST-даних, які можуть бути прийняті.
- `error_reporting`: Встановлює рівень звітування про помилки.
- `display_errors`: Встановлює, чи відображати помилки на екрані.

Перезапустіть веб-сервер: Після внесення змін у файл php.ini потрібно перезапустити веб-сервер (Apache або Nginx), щоб зміни набрали чинності.

Використовуйте директиви ini_set(): В PHP також можна використовувати функцію `ini_set()` в коді вашого сценарію для встановлення конфігураційних параметрів динамічно. Наприклад:

ini_set('memory_limit', '256M');
ini_set('max_execution_time', 60);

Це дозволяє вам змінювати параметри PHP безпосередньо в коді.

Використання .htaccess (для Apache): У веб-сервері Apache ви можете використовувати файли .htaccess для зміни конфігурації PHP для окремих каталогів або сайтів. Наприклад, ви можете встановити `php_value` або `php_flag` для встановлення конкретних параметрів. Проте, впевніться, що ви маєте дозвіл на використання .htaccess, а цей метод не рекомендується у виробничому середовищі.

Важливо пам'ятати, що зміни конфігурації PHP можуть вплинути на роботу вашого веб-додатку або сайту. Будьте обережні при зміні параметрів і перевіряйте, як вони впливають на роботу вашого коду.

Управління пам'яттю в PHP здійснюється автоматично завдяки вбудованій системі збору сміття. Основні особливості управління пам'яттю в PHP включають:

Збірка сміття: PHP використовує алгоритм зборки сміття для автоматичного визначення та вивільнення пам'яті, яку вже не використовують змінні або об'єкти. Збірка сміття автоматично відслідковує посилання на об'єкти та забезпечує вивільнення пам'яті, коли об'єкт стає недосяжним.

Ручне вивільнення пам'яті: PHP автоматично вивільняє пам'ять, коли змінні виходять з області видимості або їх значення змінюється. Однак, ви також можете використовувати функцію `unset()` для явного вивільнення пам'яті, яку використовують конкретні змінні або об'єкти.

Обмеження пам'яті: PHP дозволяє встановлювати обмеження пам'яті, доступної для виконання сценаріїв. Це робиться за допомогою параметра `memory_limit` в файлі конфігурації `php.ini`. За замовчуванням `memory_limit` має значення 128M (128 мегабайт). Якщо сценарій спробує використати більше пам'яті, ніж дозволено, буде спрацювати помилка.

Використання ресурсів: PHP дозволяє використовувати ресурси для управління зовнішніми об'єктами, такими як файлові дескриптори, підключення до баз даних, сеанси тощо. Це дозволяє звільнити пам'ять та інші ресурси, коли вони більше не потрібні.

Кешування: PHP має можливості кешування, що дозволяє зберігати результати обчислень або обробки у пам'яті або на диску. Це може покращити продуктивність та зменшити навантаження на сервер.

Важливо ретельно контролювати використання пам'яті у ваших PHP-програмах, особливо при роботі з великими обсягами даних або довготривалими процесами. Надмірне використання пам'яті може призвести до зниження продуктивності або навіть до вичерпання ресурсів сервера.

Оновлення PHP до останньої версії є важливим аспектом з точки зору безпеки, стабільності та продуктивності. Ось декілька причин, чому важливо вчасно оновлювати PHP:

Безпека: Оновлення PHP включають в себе патчі та виправлення для виявлених безпекових вразливостей. Вразливості можуть використовуватися зловмисниками для зламу або атак на ваші веб-додатки. Оновлення допомагають закрити ці вразливості та забезпечити більш безпечне середовище для вашої програми.

Виправлення помилок: Оновлення PHP також включають в себе виправлення помилок та недоліків, які можуть впливати на роботу вашого коду. Це допомагає покращити стабільність та надійність ваших веб-додатків та зменшити можливі проблеми, з якими ви можете стикнутися.

Покращення продуктивності: Кожна нова версія PHP може включати оптимізації та покращення продуктивності. Це означає, що ваші веб-додатки можуть працювати швидше та ефективніше за рахунок вдосконалень, внесених у нові версії PHP. Оновлення PHP може значно покращити продуктивність та реакцію вашого веб-сайту або додатка.

Нові функції: Кожна версія PHP включає нові функціональні можливості та розширення, які можуть бути корисними для вашого проекту. Оновлення PHP дають вам можливість використовувати останні інструменти та можливості, які допоможуть поліпшити вашу розробку та підвищити якість ваших проектів.

Сумісність з іншими технологіями: Оновлення PHP також можуть включати покращення сумісності з іншими технологіями, такими як бази даних, веб-сервери, фреймворки та інші компоненти веб-розробки. Оновлення PHP дозволяють вам

користуватися новими можливостями, які пропонуються іншими технологіями, та забезпечують сумісність з ними.

Налаштування оновлення PHP варто виконувати з регулярністю, щоб забезпечити безпеку, стабільність та продуктивність вашого веб-сервера та веб-додатків.

1.3.4. СЕРВЕРИ MYSQL / MARIADB

Для встановлення MySQL та MariaDB на Ubuntu виконайте наступні кроки:

MySQL:

Відкрийте термінал на Ubuntu.

Оновіть списки пакетів командою:

sudo apt update

Встановіть пакет MySQL Server, введіть команду:

sudo apt install mysql-server

Під час встановлення вас буде запрошено ввести пароль кореневого користувача MySQL. Введіть пароль та підтвердіть його.

Зачекайте, доки процес встановлення не завершиться. Після цього служба MySQL автоматично запуститься.

MariaDB:

Відкрийте термінал на Ubuntu.

Оновіть списки пакетів командою:

sudo apt update

Встановіть пакет MariaDB Server, введіть команду:

sudo apt install mariadb-server

Під час встановлення вас буде запрошено ввести пароль кореневого користувача MariaDB. Введіть пароль та підтвердіть його.

Зачекайте, доки процес встановлення не завершиться. Після цього служба MariaDB автоматично запуститься.

Після встановлення MySQL або MariaDB ви можете перевірити, чи служба запущена, введіть команду:

sudo systemctl status mysql # для MySQL
sudo systemctl status mariadb # для MariaDB

Тепер ви можете почати використовувати MySQL або MariaDB на своєму сервері Ubuntu.

У MySQL права доступу налаштовуються за допомогою системи керування користувачами і привілеями. Нижче наведено кілька основних кроків для налаштування прав доступу в MySQL:

Підключіться до сервера MySQL за допомогою команди:

mysql -u root -p

Введіть пароль користувача root MySQL.

Створіть нового користувача, якщо потрібно, командою:

CREATE USER 'ім'@'хост' IDENTIFIED BY 'пароль';

Де 'ім' - ім'я користувача, 'хост' - хост або IP-адреса, з якої можна здійснювати підключення (якщо ви хочете дозволити підключення з будь-якого хоста, використовуйте '%'), 'пароль' - пароль для нового користувача.

Надайте користувачу необхідні привілеї командою GRANT:

GRANT privileges ON database.table TO 'ім'@'хост';

Замість `привілеїв` вкажіть необхідні привілеї, наприклад, ALL PRIVILEGES для надання повних привілеїв. Вказуйте `database.table` для конкретної бази даних та таблиці або використовуйте * для надання прав доступу до всіх баз даних та таблиць. 'ім' та 'хост' повинні відповідати значенням, які ви використовували під час створення користувача.

Застосуйте зміни командою:

FLUSH PRIVILEGES;

Вийдіть з MySQL командою:

EXIT;

Після цих кроків користувач зазначений ім'ям отримає необхідні привілеї доступу до вказаних баз даних та таблиць. У разі потреби ви можете налаштувати додаткові привілеї або змінити існуючі привілеї для користувачів MySQL. Детальну інформацію щодо управління привілеями ви можете знайти в документації MySQL.

Вчасне оновлення MySQL має кілька важливих причин:

Безпека: Оновлення MySQL включають в себе виправлення помилок, усунення вразливостей та захисних виправлень. Зважаючи на те, що база даних є критично важливою складовою багатьох додатків, важливо забезпечувати безпеку своїх даних і запобігати можливим атакам. Оновлення допомагають уникнути використання вразливостей, які можуть бути використані зловмисниками для незаконного доступу до вашої бази даних.

Виправлення помилок та поліпшення: Оновлення MySQL також включають виправлення помилок та поліпшення функціональності. Вони можуть включати в себе оптимізацію швидкості, покращення сумісності з іншими програмними засобами, виправлення проблем, пов'язаних з роботою додатків та багато іншого. Оновлення допомагають підтримувати базу даних в оптимальному стані та забезпечувати належне функціонування системи.

Нові функції: Оновлення також можуть включати в себе нові функції та можливості. Це може означати, що ви матимете доступ до нових інструментів, які поліпшують продуктивність, розширюють можливості або спрощують розробку та адміністрування бази даних. Використання оновленого MySQL дозволяє вам скористатися всіма перевагами нових функцій.

Сумісність: Оновлення MySQL можуть бути необхідні для забезпечення сумісності з іншими компонентами вашої інфраструктури, такими як операційна система, інші програмні засоби або бібліотеки. Правильна сумісність допомагає запобігти конфліктам та забезпечити належну роботу всіх компонентів вашої системи.

Враховуючи ці причини, рекомендується вчасно оновлювати MySQL, щоб забезпечити безпеку, продуктивність та сумісність вашої бази даних. Перед оновленням рекомендується зробити резервну копію бази даних та перевірити сумісність з вашими додатками та іншими складовими системи.

Тюнінг та налаштування MySQL є важливою задачею для оптимізації продуктивності бази даних та забезпечення ефективної роботи. Ось деякі основні аспекти, які можна врахувати при тюнінгу та налаштуванні MySQL:

Конфігураційні файли: MySQL має конфігураційні файли, такі як `my.cnf` (або `my.ini` на Windows), які визначають налаштування сервера. У цих файлах можна встановити різні параметри, такі як розміри буферів, обмеження пам'яті, оптимізацію запитів тощо. Основна ідея полягає в тому, щоб налаштувати ці параметри відповідно до потреб вашого додатку та обсягу даних, щоб досягти максимальної продуктивності.

Буфери: MySQL використовує буфери для збереження даних, які використовуються під час операцій зчитування та запису. Деякі важливі буфери включають буфери кешу пам'яті (наприклад, `innodb_buffer_pool_size` для InnoDB), буфери ключового кешу (`key_buffer_size` для MyISAM) та буфери сортування (`sort_buffer_size`, `read_buffer_size`, `join_buffer_size` тощо). Правильне налаштування розмірів буферів дозволяє зменшити навантаження на дискову підсистему та покращити продуктивність.

Індексація: Відповідна індексація таблиць є важливим аспектом для швидкодії бази даних. Індекси дозволяють швидше знаходити та фільтрувати дані під час виконання запитів. Для кожної таблиці слід розглянути оптимальну індексацію, включаючи визначення первинного ключа та індексів для полів, які часто використовуються у запитах.

Оптимізація запитів: Ефективність запитів можна покращити шляхом використання оптимальних структур запитів, вибору правильних операторів, використання покриваючих індексів, уникання великих результатів запитів та використання пакетних операцій.

Моніторинг та логування: Важливо встановити моніторинг бази даних та збір логів для виявлення проблем та визначення причин погіршення продуктивності. MySQL надає різні інструменти для моніторингу, такі як Performance Schema, EXPLAIN, Slow Query Log, а також зовнішні інструменти, які можна використовувати для аналізу продуктивності.

Це лише кілька основних аспектів тюнінгу та налаштування MySQL. Оптимальне налаштування залежить від конкретних потреб вашої бази даних та додатків. Рекомендується детально ознайомитись з документацією MySQL, дослідити конкретні потреби вашого середовища та використовувати різні інструменти для моніторингу та аналізу продуктивності.

Для відлагодження та аналізу продуктивності запитів в MySQL можна використовувати команду `EXPLAIN`. Ця команда надає детальну інформацію про те, як MySQL планує та виконує запити.

Однак варто зауважити, що команда `EXPLAIN` не надає прямої інформації про реальний час виконання запиту. Щоб отримати таку інформацію, ви можете використовувати модифікатор `ANALYZE`.

Ось кілька кроків для відлагодження запитів MySQL з використанням `EXPLAIN ANALYZE`:

Відкрийте командний рядок MySQL або інтерфейс командної строки, такий як MySQL Shell або phpMyAdmin.

Введіть запит, який ви хочете відлагоджувати за допомогою `EXPLAIN ANALYZE`. Наприклад:

*EXPLAIN ANALYZE SELECT * FROM table_name WHERE column = 'value';*

Перегляньте вивід `EXPLAIN ANALYZE`, який містить детальну інформацію про спосіб виконання запиту та оцінку витрат часу. Інформація може включати такі елементи, як використані індекси, типи з'єднань, кількість прочитаних рядків, виконані операції сортування тощо.

Аналізуйте вивід, щоб ідентифікувати можливі проблеми та визначити, як ви можете покращити продуктивність запиту. Зверніть увагу на операції, які використовують багато ресурсів або займають багато часу, і розгляньте можливості оптимізації, такі як створення індексів, перерозподіл умов, оптимізація структури запиту тощо.

Застосуйте виправлення та зміни до запиту, щоб покращити його продуктивність, і повторно виконайте `EXPLAIN ANALYZE`, щоб перевірити результати.

Важливо зауважити, що використання `EXPLAIN ANALYZE` повинно бути поєднано з реальними даними та навантаженням, щоб отримати більш точну оцінку продуктивності запиту. Рекомендується також докладніше ознайомитись з документацією MySQL та дослідити інші інструменти аналізу продуктивності, які можуть бути доступні для вашого середовища.

1.4. ЗАСОБИ ВІДЛАГОДЖЕННЯ. ДЕБАГЕРИ.

При розробці на PHP можна використовувати дебагери для полегшення відлагодження та виявлення помилок у програмному коді. Дебагери допомагають крок за кроком виконувати код, досліджувати значення змінних, відстежувати виклики функцій та багато іншого. Ось кілька популярних дебагерів для PHP:

Xdebug: Xdebug є одним з найпотужніших та найбільш використовуваних дебагерів для PHP. Він надає широкий спектр функцій, таких як точкові зупинки (breakpoints), трасування стеку викликів, інтерактивне виконання, перегляд змінних та багато іншого. Xdebug може бути інтегрований з різними IDE та засобами редагування коду.

Zend Debugger: Zend Debugger є іншим потужним дебагером, який надає важливі функції, такі як точкові зупинки, трасування стеку викликів, аналіз змінних тощо. Він також може бути інтегрований з різними IDE та засобами редагування коду.

PhpStorm: PhpStorm є популярним інтегрованим середовищем розробки (IDE) для PHP, яке надає вбудовану підтримку дебагу. Ви можете використовувати PhpStorm з Xdebug або Zend Debugger для відлагодження PHP-коду. PhpStorm надає зручний інтерфейс для встановлення точок зупинки, трасування викликів, перегляду змінних, крокування по коду тощо.

Visual Studio Code: Visual Studio Code (VS Code) є популярним легким редактором коду, який також підтримує дебаг PHP. Ви можете використовувати розширення, такі як PHP Debug, для встановлення точок зупинки, трасування викликів та відлагодження PHP-коду безпосередньо у VS Code.

Це лише кілька прикладів дебагерів та інструментів, які можна використовувати для розробки на PHP. Важливо підібрати інструмент, який найкраще відповідає вашим потребам та платформі розробки. Документація та ресурси кожного дебагера надають докладні інструкції щодо налаштування та використання для відлагодження PHP-коду.

Щоб встановити та налаштувати Xdebug на Ubuntu, виконайте наступні кроки:

Встановіть пакет `php-xdebug`, виконавши наступну команду в терміналі:

sudo apt install php-xdebug

Відкрийте файл налаштувань Xdebug для редагування. Наприклад, для PHP 7.4 відредагуйте файл `/etc/php/7.4/mods-available/xdebug.ini` за допомогою текстового редактора:

sudo nano /etc/php/7.4/mods-available/xdebug.ini

Додайте наступні рядки в кінець файлу `xdebug.ini`:

zend_extension=/usr/lib/php/20190902/xdebug.so
xdebug.mode=debug
xdebug.start_with_request=trigger
xdebug.client_port=9000

Збережіть зміни та закрийте файл.
Перезапустіть веб-сервер або процес PHP-FPM, щоб застосувати зміни:

sudo service apache2 restart # якщо використовуєте Apache
sudo service php7.4-fpm restart # якщо використовуєте PHP-FPM

Переконайтеся, що Xdebug встановлений та налаштований, виконавши команду:

php -i | grep xdebug

Ви повинні побачити вивід, що підтверджує наявність Xdebug та його поточні налаштування.

Для використання Xdebug з редактором коду або IDE, налаштуйте його для підключення до дебагера Xdebug на порті 9000.

Це базові кроки для встановлення та налаштування Xdebug на Ubuntu. Врахуйте, що шлях до `xdebug.so` та версія PHP може відрізнятися в залежності від вашої конфігурації. Рекомендується перевірити офіційну документацію Xdebug для

отримання докладнішої інформації та додаткових налаштувань, які можуть бути потрібні у вашому випадку.

Профайлер в PHP - це інструмент, який дозволяє аналізувати продуктивність вашого PHP-коду та визначати місця, де відбувається найбільше витрат часу. Він допомагає ідентифікувати потенційні проблеми та оптимізувати швидкодію програми. Ось кілька популярних профайлерів для PHP:

Xdebug: Xdebug, крім відлагодження, також має функціональність профілювання. Ви можете використовувати Xdebug для збору профілювальних даних, таких як час виконання, кількість викликів функцій, трасування стеку викликів тощо. Вивід профілювальних даних може бути аналізований за допомогою інструментів, таких як KCacheGrind або WinCacheGrind.

XHProf: XHProf є інструментом профілювання, розробленим командою Facebook. Він надає докладну інформацію про використання CPU, пам'яті та функцій вашого PHP-коду. XHProf можна налаштувати як розширення PHP або якій-небудь серверний демон. Результати профілювання можуть бути переглянуті за допомогою веб-інтерфейсу XHProf.

Blackfire: Blackfire є інструментом профілювання та аналізу продуктивності, який надає детальну інформацію про витрати ресурсів та проблеми у вашому PHP-коді. Він пропонує глибокий аналіз швидкодії програми та може інтегруватись з різними середовищами розробки.

Tideways: Tideways - це інструмент профілювання PHP, який дозволяє аналізувати продуктивність вашого коду та знаходити місця, де витрачається найбільше часу. Він надає глибоку інформацію профункції, запити до бази даних, HTTP-виклики тощо. Tideways пропонує веб-інтерфейс для аналізу результатів профілювання.

Це лише деякі приклади профайлерів для PHP, які доступні для використання. Кожен з них має свої особливості та може надавати різні типи даних для аналізу. Вибір підходящого профайлера залежить від ваших потреб та вподобань. Рекомендується ознайомитись з документацією та ресурсами кожного інструменту, щоб отримати більш детальну інформацію та налаштувати його для вашого середовища розробки.

Для встановлення XHProf на Ubuntu виконайте наступні кроки:

Встановіть пакети `graphviz` та `php-dev` за допомогою наступної команди:

sudo apt install graphviz php-dev

Завантажте XHProf з репозиторію GitHub:

git clone https://github.com/phacility/xhprof.git

Перейдіть до папки xhprof:

cd xhprof

Змініть доцільність файлу `xhprof_lib/config.php`, розкоментувавши рядок з `$XHProfLib['display']` та встановивши значення `1`:

cp xhprof_lib/config.sample.php xhprof_lib/config.php
nano xhprof_lib/config.php

Скомпілюйте розширення XHProf:

phpize
./configure
make
sudo make install

Створіть файл налаштувань для XHProf у директорії PHP-модулів:

sudo nano /etc/php/7.4/mods-available/xhprof.ini

Додайте наступний вміст у файл `xhprof.ini`:

extension=xhprof.so
xhprof.output_dir=/var/www/html/xhprof # замініть шлях на бажаний

Збережіть файл та закрийте його.

Створіть символічне посилання на `xhprof.ini` в папці `conf.d`, щоб PHP автоматично завантажував розширення:

sudo ln -s /etc/php/7.4/mods-available/xhprof.ini /etc/php/7.4/cli/conf.d/20-xhprof.ini

sudo ln -s /etc/php/7.4/mods-available/xhprof.ini /etc/php/7.4/apache2/conf.d/20-xhprof.ini

Перезапустіть веб-сервер та процес PHP-FPM для застосування змін:

sudo service apache2 restart # якщо використовуєте Apache

sudo service php7.4-fpm restart # якщо використовуєте PHP-FPM

Тепер XHProf має бути встановлений та налаштований на вашому сервері Ubuntu. Ви можете використовувати його для профілювання та аналізу продуктивності свого PHP-коду. Докладнішу інформацію про використання XHProf ви знайдете у документації, яка постачається разом з XHProf.

Питання для самоконтролю:

1. Для чого призначені Apache та Nginx?
2. Як встановити PHP?
3. Чому важливо вчасно оновлювати PHP?
4. Як встановити MySQL?
5. Як відлагоджувати запити?

1.5. ОГЛЯД СИСТЕМ ІНТЕГРОВАНОЇ РОЗРОБКИ ВЕБ-ДОДАТКІВ

Інтегровані середовища розробки (IDE) є потужними інструментами для розробки PHP-програм. Вони надають широкий набір функцій та можливостей для полегшення процесу розробки, таких як автодоповнення коду, налагоджування, керування проектами та багато іншого. Ось кілька популярних IDE для розробки на PHP:

PhpStorm: PhpStorm від компанії JetBrains є одним з найпопулярніших IDE для розробки на PHP. Він пропонує повний набір функцій, включаючи автодоповнення, відлагодження, керування версіями, вбудовані інструменти для роботи з базами даних та інші корисні інструменти. PhpStorm також підтримує плагіни для інтеграції з іншими інструментами розробки та фреймворками PHP.

Visual Studio Code: Visual Studio Code (VS Code) є легким редактором коду, який також надає потужні можливості розробки на PHP завдяки великій кількості розширень. VS Code має вбудовану підтримку відлагодження, автодоповнення, керування проектами та інші корисні функції. Він також підтримує інтеграцію з Git та іншими популярними інструментами.

NetBeans: NetBeans - це безкоштовне інтегроване середовище розробки, яке підтримує PHP. Воно пропонує широкий спектр функцій, таких як автодоповнення, відлагодження, підтримка фреймворків, керування проектами та інші. NetBeans має дружній інтерфейс та вбудовані інструменти для розробки PHP-програм.

Eclipse PDT: Eclipse PDT (PHP Development Tools) є розширенням для платформи Eclipse, призначеним для розробки PHP-програм. Він надає функції, такі як автодоповнення, відлагодження, керування проектами та підтримку фреймворків. Eclipse PDT також підтримує інтеграцію з іншими інструментами розробки та має широкі можливості налаштування.

Це лише кілька прикладів IDE для розробки на PHP. Кожен з них має свої особливості та може відповідати різним потребам розробників. Рекомендується спробувати кілька IDE та обрати той, який найбільше відповідає вашому стилю розробки та потребам.

Щоб встановити PhpStorm на Ubuntu, виконайте наступні кроки:

Завантажте PhpStorm з офіційного сайту JetBrains (https://www.jetbrains.com/phpstorm/download/). Зазвичай вони надають файл `.tar.gz` для Linux.

Розпакуйте завантажений архів в потрібну вам папку. Наприклад, використовуючи команду `tar`, розпакуйте архів до папки `/opt`:

tar -xzf PhpStorm-.tar.gz -C /opt*

Перейдіть до розпакованої папки PhpStorm:

*cd /opt/PhpStorm-**

Запустіть PhpStorm з використанням скрипту `phpstorm.sh`:

./bin/phpstorm.sh

При запуску ви побачите вікно встановлення PhpStorm. Виберіть опцію "Do not import settings" (не імпортувати налаштування) або встановіть налаштування за потреби.

Прочитайте та прийміть ліцензійну угоду.

PhpStorm запуститься та почне ініціалізацію. Це може зайняти деякий час.

Після ініціалізації ви побачите вікно вітання PhpStorm.

Тепер PhpStorm повинен бути успішно встановлений на вашій системі Ubuntu. Ви можете запустити його з командного рядка за допомогою команди `phpstorm` або з меню додатків вашої системи. Після першого запуску PhpStorm попросить вас ввести ліцензійний ключ або вибрати пробний режим.

Після встановлення PhpStorm на Ubuntu виникає декілька кроків налаштування, щоб забезпечити оптимальну роботу IDE. Ось кілька важливих кроків налаштування:

Встановіть PHP: Переконайтеся, що ви маєте встановлену PHP на своєму сервері. PhpStorm використовує PHP для виконання коду та надання автодоповнень та перевірок. Встановіть PHP та його необхідні розширення за допомогою команди `sudo apt install php` або іншим способом, який ви обираєте.

Налаштування веб-сервера: PhpStorm потребує налаштування веб-сервера для правильного відображення та відлагодження веб-проектів. Виберіть веб-сервер, який ви використовуєте (наприклад, Apache або Nginx) і переконайтеся, що він налаштований правильно для вашого проекту.

Налаштування PHP Interpreter: Встановіть шлях до встановленого PHP-інтерпретатора у PhpStorm. Для цього виберіть "File" (Файл) у верхньому меню, потім "Settings" (Налаштування). У вікні налаштувань перейдіть до "Languages & Frameworks" (Мови та фреймворки) та виберіть "PHP". На вкладці "PHP" встановіть шлях до встановленого PHP-інтерпретатора.

Налаштування бази даних: Якщо ви працюєте з базою даних, налаштуйте з'єднання з базою даних у PhpStorm. Виберіть "File" (Файл) у верхньому меню, потім "Settings" (Налаштування). У вікні налаштувань перейдіть до "Database" (База даних) та додайте нове з'єднання з вашою базою даних, вказавши необхідні налаштування підключення.

Налаштування автодоповнень та форматування: PhpStorm надає потужні функції автодоповнень та форматування коду. Ви можете налаштувати їх під свої потреби у вікні налаштувань "Editor" (Редактор) та "Code Style" (Стиль коду).

Налаштування клавішних скорочень: PhpStorm має велику кількість клавішних скорочень для прискорення роботи. Ви можете налаштувати свої власні клавішні скорочення у вікні налаштувань "Keymap" (Клавіатура).

Це лише декілька основних кроків налаштування PhpStorm для Ubuntu. PhpStorm має багато інших налаштувань та можливостей, які ви можете дослідити для власних потреб. Документація PhpStorm містить більш детальну інформацію про всі налаштування та функціональність IDE.

Питання для самоконтролю:

1. Які IDE доступні для використання при розробці на PHP?
2. Як встановити PHPStorm на Ununtu?
3. Як налаштувати PHPStorm на Ununtu?

2. ОСНОВИ РОБОТИ З CMF DRUPAL (SITE BUILDING)

Drupal 8 - це потужний фреймворк для розробки веб-сайтів та додатків. Є багато причин, чому варто використовувати Drupal для розробки веб-сайту або додатку:

Гнучкість та розширюваність: Drupal надає широкі можливості для розробки різноманітних веб-сайтів та додатків. Він має модульну архітектуру, що дозволяє легко додавати нові функціональність за допомогою модулів. З великим розмаїттям модулів, доступних у спільноті Drupal, ви можете знайти рішення для багатьох ваші потреб.

Складність проектів: Drupal може бути використаний для розробки як невеликих особистих веб-сайтів, так і складних корпоративних порталів або електронних комерційних платформ. Він має гнучкі інструменти та можливості для розробки проектів різної складності і масштабу.

Спільнота та підтримка: Drupal має велику та активну спільноту розробників і користувачів. Це означає, що є багато ресурсів, документації, форумів та інших засобів підтримки, на які ви можете покластися. Ви можете отримати допомогу, поради та відповіді на ваші питання від інших членів спільноти Drupal.

Безпека: Drupal має добре встановлену систему безпеки. Його безпековий комітет постійно моніторить потенційні загрози та випускає оновлення для захисту від відомих вразливостей. Drupal також має вбудовану систему контролю доступу та інші інструменти безпеки, що дозволяють захистити ваш веб-сайт від зловмисників.

Міжнародна підтримка: Drupal надає підтримку для багатомовних веб-сайтів і локалізації. Ви можете легко створювати веб-сайти, що підтримують різні мови та регіони.

Загалом, Drupal є потужним і гнучким рішенням для розробки веб-сайтів та додатків. Він має широкий функціонал, активну спільноту та добру підтримку безпеки. Вибір використання Drupal залежить від ваших конкретних потреб, проекту та навичок розробника.

2.1. ЗАГАЛЬНА ХАРАКТЕРИСТИКА СИСТЕМИ

Розглянемо кілька основних рис і можливостей Drupal 8.

Модульна архітектура: Drupal 8 побудований на модульній архітектурі, що дозволяє розширювати його функціональність за допомогою модулів. У Drupal 8 існує велика кількість модулів, які можна використовувати для розширення функціональності вашого веб-сайту.

Гнучкість та масштабованість: Drupal 8 надає гнучкі інструменти для розробки веб-сайтів різних розмірів та складності. Він може обробляти невеликі особисті веб-сайти, корпоративні портали та навіть складні електронні комерційні платформи.

Мультисайтовість: Drupal 8 підтримує мультисайтовість, що означає, що ви можете використовувати один екземпляр Drupal для керування кількома веб-сайтами. Це зручно для управління кількома проектами з одним інсталятором Drupal.

Модерні технології: Drupal 8 використовує сучасні технології, такі як Symfony Framework, Twig шаблони та PSR-стандарти. Це робить його більш сучасним і підтримує розробку за допомогою найкращих практик.

Мобільна підтримка: Drupal 8 надає підтримку для розробки мобільних веб-сайтів та додатків. Він має вбудовану адаптивність та може надавати оптимізований вигляд для різних пристроїв.

Керування контентом: Drupal 8 надає потужні можливості керування контентом, включаючи створення, редагування та публікацію контенту. Ви можете легко організовувати контент у вигляді статей, блогів, сторінок, зображень, відео та інших типів контенту.

Спільнота та підтримка: Drupal має велику спільноту розробників та користувачів, які активно співпрацюють та надають підтримку одне одному. Є багато ресурсів, документації, форумів та інших засобів, щоб отримати допомогу та поради з використання Drupal 8.

Drupal 8 є потужним і гнучким рішенням для розробки веб-сайтів та додатків. Він надає широкий спектр функціональності, що дозволяє розробникам створювати різноманітні проекти з високою продуктивністю та масштабованістю.

Наразі у використанні перебувають Drupal 7 і Drupal 8 - це дві версії популярної системи управління контентом (CMS) Drupal. Ось деякі ключові різниці між Drupal 7 і Drupal 8:

Архітектура: Drupal 7 використовує свою власну архітектуру, включаючи систему гаків (hooks) і функції, які розширюють його функціональність. Drupal 8 базується на Symfony Framework і використовує сучасну архітектуру, що дозволяє використовувати більшу кількість стандартів та компонентів з Symfony.

Об'єктно-орієнтоване програмування: Drupal 7 використовує процедурний підхід до програмування, тоді як Drupal 8 підтримує об'єктно-орієнтоване програмування. Це спрощує розробку, підтримку і розширення Drupal 8.

Мобільна підтримка: Drupal 7 не має вбудованої підтримки для адаптивного дизайну та розробки мобільних додатків. У Drupal 8 побудована мобільна підтримка, що дозволяє створювати мобільні веб-сайти та додатки з адаптивним дизайном.

Удосконалені редактори контенту: Drupal 8 має вдосконалені редактори контенту, такі як WYSIWYG-редактор CKEditor, який дозволяє користувачам зручно форматувати та створювати контент.

Покращена міжнародна підтримка: Drupal 8 має покращену підтримку багатомовності та локалізації. Це дозволяє легко створювати веб-сайти, що підтримують різні мови та регіони.

Удосконалене керування конфігурацією: Drupal 8 надає нову систему керування конфігурацією, яка дозволяє розробникам керувати налаштуваннями сайту у зручний спосіб та зберігати їх у кодовому репозиторії.

Покращена екосистема модулів: Drupal 8 має нову систему модулів, яка полегшує розробку, розповсюдження та підтримку модулів. Багато модулів було переписано або оновлено для сумісності з Drupal 8.

Це лише кілька різниць між Drupal 7 і Drupal 8. Обидві версії мають свої переваги і можуть бути використані для розробки різноманітних веб-сайтів та додатків. Вибір між ними залежить від потреб, знань і вимог проекту.

2.2. ІНСТАЛЯЦІЯ ТА БАЗОВІ НАЛАШТУВАННЯ

Ми рекомендуємо використовувати DDEV для локальної розробки на Drupal з наступних причин:

Легко встановити та налаштувати: DDEV пропонує простий та швидкий процес встановлення і налаштування. Ви можете швидко налаштувати новий проект Drupal і почати працювати з ним всього за декілька кроків.

Незалежність від середовища: DDEV використовує Docker для створення контейнерів, що дозволяє вам створювати ізольоване середовище для вашого проекту. Ви можете бути впевнені, що ваша розробка відбувається відповідно до встановлених налаштувань і не впливає на інші проекти або вашу систему в цілому.

Конфігурація проекту: DDEV дозволяє легко налаштувати ваш проект, включаючи налаштування веб-сервера, бази даних, PHP-версії та інші параметри. Ви можете зберігати конфігурацію проекту в репозиторії, що дозволяє всім учасникам команди мати однакове середовище розробки.

Автоматичне керування середовищем: DDEV автоматично налаштовує веб-сервер, базу даних та інші компоненти для вас. Ви можете легко запускати, зупиняти та перезапускати ваше середовище розробки з командного рядка або за допомогою графічного інтерфейсу.

Інтеграція з іншими інструментами: DDEV інтегрується з популярними інструментами розробки, такими як Composer, Drush, Xdebug та інші. Ви можете використовувати ці інструменти для розширення функціональності та полегшення процесу розробки на Drupal.

Підтримка і спільнота: DDEV має активну спільноту, яка надає допомогу, документацію та розширення для інструменту. Ви можете швидко знайти відповіді на свої питання та отримати підтримку у разі потреби.

Загалом, використання DDEV для локальної розробки на Drupal дозволяє вам швидко налаштувати зручне та ізольоване середовище для вашого проекту. Він полегшує процес розробки та сприяє збільшенню продуктивності.

Для встановлення DDEV на Ubuntu виконайте наступні кроки:

Відкрийте термінал на вашому Ubuntu.

Встановіть необхідні залежності, якщо їх ще немає, виконавши наступну команду:

sudo apt-get install curl git docker docker-compose

Виконайте команду для завантаження та встановлення DDEV:

curl -L https://github.com/drud/ddev/releases/latest/download/ddev_linux -o ddev
chmod +x ddev
sudo mv ddev /usr/local/bin

Перевірте, чи DDEV встановився правильно, виконавши команду:

ddev version

Ви повинні побачити вихідну інформацію про версію DDEV.

Тепер DDEV повинен бути успішно встановлений на вашому Ubuntu. Ви можете перейти до вашого проекту Drupal та використовувати команду `ddev` для управління локальним середовищем розробки.

Щоб створити чистий проект на DDEV для Drupal 8, слід виконати такі кроки:

Відкрийте термінал на вашому Ubuntu.

Перейдіть до каталогу, в якому ви хочете створити новий проект Drupal.

Виконайте наступну команду для створення нового проекту:

ddev config --project-type=drupal8 --docroot=web --create-docroot

Ця команда створить новий проект з директорією `web` як кореневою директорією проекту Drupal 8. Опція `--create-docroot` створить директорію `web`, якщо вона ще не існує.

Після створення проекту запустіть команду:

ddev start

Ця команда запустить DDEV та налаштує локальне середовище розробки для вашого проекту Drupal 8.

Відкрийте веб-браузер і перейдіть за URL-адресою, яку вказує DDEV після успішного запуску. Наприклад, `http://projectname.ddev.local`, де `projectname` - це назва вашого проекту.

Тепер у вас є чистий проект Drupal 8, налаштований з використанням DDEV на вашому локальному середовищі. Ви можете продовжити розробку вашого проекту, встановлюючи модулі, налаштовуючи теми та розробляючи новий функціонал.

Коли ми маємо локальне оточення - можемо розпочати процес встановлення Drupal 8.

Використання DDEV і нативного оточення для розробки на Drupal має свої переваги та обмеження. Ось кілька аспектів порівняння:

Установка та налаштування: DDEV пропонує простий та швидкий процес встановлення та налаштування. Він використовує Docker для створення ізольованого середовища, що дозволяє легко встановлювати та управляти залежностями. Нативне оточення потребує окремого налаштування веб-сервера, бази даних, PHP та інших компонентів.

Портативність та ізоляція: DDEV забезпечує ізольоване середовище розробки, що дозволяє легко переміщати ваш проект між різними системами. Ви можете легко поділитися проектом з іншими розробниками або розгорнути його на продакшн сервері. Нативне оточення зазвичай прив'язане до конкретної системи та потребує додаткових налаштувань для переміщення проекту.

Швидкість та продуктивність: DDEV може бути швидшим у розгортанні та змінах середовища, оскільки він використовує контейнери Docker, які можуть бути легко перезапущені та оновлені. Нативне оточення може вимагати більше часу та зусиль для налаштування і зміни компонентів.

Флексібільність: Нативне оточення надає більшу гнучкість в налаштуванні компонентів та розширення функціональності. Ви можете встановлювати різні версії PHP, настроювати веб-сервер за своїми потребами та використовувати будь-які інші компоненти, які вам потрібні. DDEV надає певну структуру та конфігурацію, яку можна змінювати, але з обмеженою гнучкістю.

Спільнота та підтримка: DDEV має активну спільноту розробників, яка надає підтримку та документацію. Нативне оточення може мати більш широку спільноту, оскільки воно базується на відкритих стандартах та інструментах.

При виборі між DDEV та нативним оточенням, враховуйте власні потреби та вміння. DDEV може бути більш простим та швидким варіантом для швидкої розробки та спільної роботи, тоді як нативне оточення може надати більшу гнучкість та контроль над середовищем.

2.2.1. СИСТЕМНІ ВИМОГИ

Системні вимоги для Drupal 8:

Веб-сервер: Рекомендовано використовувати веб-сервер Apache або Nginx. Мінімальною версією Apache є 2.0 або вище, а Nginx - 1.9.0 або вище.

PHP: Drupal 8 вимагає PHP версії 7.3 або вище. Рекомендовано використовувати PHP 7.4 або PHP 8 для найкращої сумісності та продуктивності. Потрібні такі PHP-розширення: pdo, mysqlnd, curl, gd, mbstring, openssl, xml, zip та інші.

База даних: Drupal 8 підтримує різні системи управління базами даних (СУБД), такі як MySQL/MariaDB, PostgreSQL та SQLite. Рекомендовано використовувати MySQL версії 5.7 або вище, або MariaDB версії 10.3 або вище.

Операційна система: Drupal 8 може бути встановлений на різних операційних системах, включаючи Linux, Windows та macOS. Однак, важливо враховувати сумісність зі зазначеними версіями веб-сервера, PHP та бази даних для вибраної операційної системи.

Додаткові залежності: Для певних функцій та розширень Drupal 8 може вимагати додаткові залежності. Наприклад, для обробки зображень та створення зменшених копій, може знадобитися наявність бібліотеки ImageMagick або GD.

Це загальні системні вимоги для Drupal 8, і реальні вимоги можуть варіюватися в залежності від конкретних потреб і налаштувань вашого проекту. Рекомендується перевірити документацію Drupal для детальнішої інформації та актуальних вимог.

Для стабільної роботи рекомендується наявність як мінімум 512 Mb пам’яті для використання PHP.

Використання дебагерів та відладчиків пов’язане із суттєвим навантаженням на апаратну складову, тому для комфортної розробки рекомендується використовувати наступні системні вимоги до апаратної складової:

- CPU: Intel Core i5 або аналонічний
- RAM: 16 Gb
- HDD/SSD (рекомендовано) з 40 Gb вільного простору

2.2.2. НАЛАШТУВАННЯ ЛОКАЛЬНОГО ОТОЧЕННЯ

DDEV надає можливості налаштування для вашого проекту Drupal. Ось декілька аспектів налаштування DDEV:

Конфігураційний файл ddev.yaml: DDEV використовує конфігураційний файл `ddev.yaml` для налаштування проекту. Цей файл знаходиться в кореневій директорії вашого проекту і містить параметри, такі як назва проекту, версія PHP, тип веб-сервера, база даних та інші. Ви можете налаштувати ці параметри відповідно до вашої потреби.

Доступні сервіси: DDEV надає можливість налаштування різних сервісів для вашого проекту, таких як веб-сервер, база даних, кешування, електронна пошта тощо. Ви можете вибрати певні версії сервісів, налаштувати порти, обмеження ресурсів тощо.

Мережі та домени: DDEV автоматично створює окрему мережу Docker для вашого проекту і надає домени в форматі `projectname.ddev.local`. Проте, ви можете налаштувати власні домени або SSL-сертифікати, якщо потрібно.

Додаткові конфігураційні файли: Крім `ddev.yaml`, ви також можете використовувати додаткові конфігураційні файли для налаштування окремих сервісів

або зберігання інших параметрів. Наприклад, ви можете використовувати `php.ini` для налаштування PHP або `drupal.settings.php` для специфічних налаштувань Drupal.

Команди ddev: DDEV надає набір команд, які ви можете використовувати для управління вашим проектом. Ви можете запускати, зупиняти, перезапускати сервіси, виконувати команди Drush або консоль Symfony, і багато іншого.

Це лише деякі аспекти налаштування DDEV. Загальною ідеєю є те, що ви можете налаштувати DDEV згідно з вашими потребами та специфічними вимогами вашого проекту Drupal. Документація DDEV містить детальнішу інформацію про різні параметри та опції налаштування.

Основні команди DDEV дозволяють вам керувати та взаємодіяти з вашим проектом Drupal. Ось кілька основних команд DDEV:

- ddev start: Запускає ваше середовище розробки. Ця команда створює та запускає Docker-контейнери для веб-сервера, бази даних та інших сервісів, відповідно до налаштувань в `ddev.yaml`. Вона також встановлює необхідні залежності та запускає ваш проект Drupal.
- ddev stop: Зупиняє ваше середовище розробки. Ця команда зупиняє контейнери, що виконуються для веб-сервера, бази даних та інших сервісів.
- ddev restart: Перезапускає ваше середовище розробки. Ця команда зупиняє та знову запускає контейнери для оновлення налаштувань або застосування змін.
- ddev describe: Виводить детальну інформацію про ваше середовище розробки, включаючи URL-адреси доступу, налаштування сервісів та інше.
- ddev ssh: Запускає командний рядок SSH в контейнері вашого проекту. Ви можете використовувати цю команду для виконання команд або отримання доступу до файлів вашого проекту.
- ddev exec: Виконує команду всередині контейнера вашого проекту. Ви можете використовувати цю команду для виконання команд Drush, Composer або будь-яких інших команд, необхідних для вашого проекту.
- ddev import-db: Імпортує резервну копію бази даних в ваш проект. Ви можете використовувати цю команду для відновлення бази даних з резервної копії.

- ddev export-db: Експортує базу даних вашого проекту в резервну копію. Ви можете використовувати цю команду для створення резервної копії бази даних.

Це лише кілька основних команд DDEV. Ви можете отримати більш докладну інформацію про всі команди та їх параметри, виконавши команду `ddev help`.

DDEV дозволяє налаштовувати змінні оточення для вашого проекту Drupal. Змінні оточення використовуються для зберігання конфіденційних або залежних від середовища значень, таких як налаштування бази даних, ключі API, налаштування електронної пошти тощо. Ось як налаштувати змінні оточення в DDEV:

Відкрийте файл `ddev.yaml` в кореневій директорії вашого проекту.

Додайте розділ `environment_variables` до вашого файлу `ddev.yaml`. Наприклад:

```
version: '1.16.2'
name: myproject
type: drupal8

environment_variables:
  DB_NAME: mydatabase
  DB_USER: myusername
  DB_PASSWORD: mypassword
```

Додайте змінні оточення, які вам потрібні, і їх значення. Назви змінних і значення можуть бути визначені відповідно до ваших потреб.

Після збереження файлу `ddev.yaml` перезапустіть ваше середовище розробки, виконавши команду `ddev restart`.

Ваші змінні оточення тепер доступні у вашому проекті Drupal через змінну `$_ENV`. Наприклад, для отримання значення `DB_NAME` в PHP, ви можете використовувати:

```
$dbName = $_ENV['DB_NAME'];
```

Таким чином, ви можете зберігати конфіденційні дані та інші налаштування в змінних оточення DDEV, що полегшує конфігурування та переносимість проекту між середовищами.

2.2.3. ІНСТАЛЯЦІЯ

Для встановлення Drupal 8 в DDEV слід виконати наступні кроки:

Відкрийте термінал та перейдіть до директорії, де ви хочете створити свій проект Drupal.

Запустіть наступну команду, щоб створити новий проект Drupal 8:

ddev config --project-type=drupal8 --docroot=web --create-docroot

Ця команда створює новий проект з кореневою директорією `web`, яка є стандартною для Drupal 8. Опція `--create-docroot` створює директорію `web`, якщо вона ще не існує.

Після створення проекту запустіть команду:

ddev start

Ця команда запускає DDEV та налаштовує локальне середовище розробки для вашого проекту Drupal 8.

Відкрийте веб-браузер і перейдіть за URL-адресою, яку вказує DDEV після успішного запуску. Наприклад, `http://projectname.ddev.local`, де `projectname` - це назва вашого проекту.

Виберіть мову та налаштування сайту на початковій сторінці установки Drupal. Слідуйте інструкціям на екрані для завершення установки.

Тепер ви маєте Drupal 8 успішно встановлене та працюєте на вашому локальному середовищі DDEV. Ви можете продовжити налаштування та розробку вашого проекту.

Для розгортання проекту Drupal 8 з використанням Composer слід виконати наступні кроки:

Встановлення Composer: Перш ніж розгортати Drupal 8 з використанням Composer, переконайтеся, що у вас встановлений Composer на вашому комп'ютері. Ви можете завантажити та встановити Composer з офіційного сайту (https://getcomposer.org).

Створення проекту Drupal 8: Відкрийте командний рядок (термінал) та перейдіть до директорії, де ви хочете створити ваш проект Drupal 8.

Запустіть наступну команду для створення проекту Drupal 8 з використанням Composer:

composer create-project drupal/recommended-project:8.x projectname

У цій команді `projectname` - це назва вашого проекту. Вона буде використовуватись для створення директорії проекту та завантаження коду Drupal 8.

Після завершення команди Composer буде створена нова директорія з назвою `projectname`. Ця директорія міститиме весь необхідний код Drupal 8.

Налаштування бази даних: Створіть нову базу даних для вашого проекту Drupal 8 на вашому сервері баз даних.

Конфігурація файла налаштувань: Скопіюйте файл `sites/default/default.settings.php` та перейменуйте його у `sites/default/settings.php`. Відредагуйте цей файл та вкажіть налаштування бази даних, такі як ім'я бази даних, користувач, пароль тощо.

Установка Drupal 8: Відкрийте браузер та перейдіть до URL-адреси вашого проекту Drupal 8. Слідуйте інструкціям на екрані для завершення процесу установки. Введіть необхідну інформацію, включаючи назву сайту, обліковий запис адміністратора та інше.

Після успішного завершення установки ви зможете використовувати свій проект Drupal 8.

Це загальний підхід до розгортання Drupal 8 з використанням Composer. Ви можете налаштувати його додатково залежно від ваших потреб та конфігурації сервера.

2.2.4. ВИКОРИСТАННЯ ІНТЕРПРЕТАТОРА DRUSH

Drush (Drupal Shell) - це набір командного рядка для керування та автоматизації різних завдань в Drupal. Він надає розширені можливості для управління сайтом Drupal безпосередньо з командного рядка, що полегшує розробку, адміністрування та підтримку Drupal-проектів. Ось декілька основних функцій та можливостей Drush:

Управління модулями та темами: Drush дозволяє встановлювати, відключати, активувати, видаляти та оновлювати модулі та теми Drupal. Ви можете виконувати ці дії з командного рядка замість адміністративного інтерфейсу.

Виконання команд бази даних: Drush надає можливість виконувати команди бази даних, такі як імпорт, експорт, створення резервної копії, відновлення та оптимізація бази даних. Це полегшує управління та роботу з базою даних Drupal.

Виконання команд генерації коду: Drush дозволяє генерувати код автоматично для різних сутностей Drupal, таких як модулі, теми, блоки, форми та інші. Ви можете швидко створювати шаблони коду, що полегшує розробку.

Виконання тестів та планування завдань: Drush надає можливість виконувати тести модулів Drupal, запускати тестові сценарії та планувати автоматичні завдання за допомогою крону. Це спрощує тестування та автоматизацію певних процесів.

Управління конфігурацією: Drush дозволяє імпортувати та експортувати конфігурацію Drupal між різними середовищами. Ви можете керувати конфігурацією вашого сайту з командного рядка та забезпечити однорідність середовищ.

Drush є потужним інструментом для розробників та адміністраторів Drupal, який прискорює та автоматизує багато повсякденних завдань управління Drupal-проектами. Він допомагає знизити час, затрачений на повторювані дії та полегшує роботу з Drupal.

Для встановлення Drush слід виконати наступні кроки:

Перевірте вимоги: Переконайтеся, що ваша система відповідає вимогам Drush. Drush потребує PHP версії 7.2 або вище, а також певних залежностей, таких як Composer та розширення PHP (зокрема, mbstring та zip). Переконайтеся, що ці вимоги виконуються на вашій системі.

Встановлення Composer: Якщо у вас ще немає Composer на вашій системі, встановіть його. Ви можете завантажити та встановити Composer з офіційного сайту (https://getcomposer.org).

Встановлення Drush: Відкрийте командний рядок (термінал) та виконайте наступну команду для встановлення Drush за допомогою Composer:

composer global require drush/drush

Ця команда встановить Drush глобально на вашу систему.

Додавання шляху до Drush: Після встановлення Drush додайте шлях до Drush до вашого файлу `~/.bashrc` або `~/.bash_profile`. Відкрийте файл у вашому редакторі та додайте наступний рядок:

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

Збережіть файл та закрийте його.

Оновлення залежностей: Виконайте наступну команду, щоб оновити залежності Drush:

composer global update

Перевірте встановлення: Після завершення встановлення перезапустіть командний рядок та виконайте команду `drush --version`, щоб перевірити, чи встановлено Drush на вашу систему. Ви повинні побачити вивід з номером версії Drush.

Тепер Drush повинен бути встановлений та готовий до використання на вашій системі. Ви можете використовувати різні команди Drush для управління та автоматизації вашого проекту Drupal.

Drush має кілька основних версій, таких як Drush 7, Drush 8 та Drush 9/10. Ось загальна розбіжність між цими версіями:

Drush 7: Це була попередня версія Drush, яка підтримується лише для Drupal 7 проектів. Вона має базові функції для управління модулями, темами, командами бази даних та іншими аспектами Drupal 7.

Drush 8: Drush 8 був розроблений для підтримки як Drupal 7, так і Drupal 8 проектів. Він вводить нові функції та покращення, такі як підтримка модулів Composer, покращений шаблон коду, більш потужні команди генерації та інші функції, які полегшують розробку та управління Drupal-проектами.

Drush 9/10: Drush 9 є зміненою версією, яка більше не підтримує Drupal 7, але повністю підтримує Drupal 8. Деякі з основних змін у Drush 9/10 включають перехід до Symfony Console для керування командами, сумісність з Composer 2, покращену підтримку конфігурації та інші функції.

Основна різниця між Drush 8 та Drush 9/10 полягає у переході до Symfony Console, що полегшує розширення та використання команд. Drush 9/10 також використовує підхід, що сприяє більшій стабільності та сумісності з сучасними версіями Drupal.

Вибір версії Drush залежить від версії Drupal, яку ви використовуєте. Для Drupal 7 вам потрібен Drush 7, для Drupal 8 - Drush 9/10. Важливо оновлювати Drush до останньої стабільної версії, щоб отримати найкращу підтримку та нові функції.

Drush - це могутній набір командного рядка для управління та автоматизації Drupal. Ось детальний опис всіх основних команд Drush та їх призначення:

- archive-dump: Створює архівну копію вашого Drupal-сайту. Ця команда дозволяє вам зробити резервну копію вашого сайту для подальшого відновлення.
- cache-clear: Очищає кеш Drupal. Кеш зберігає тимчасові дані, які прискорюють роботу вашого сайту. Очищення кешу може бути корисним, коли ви внесли зміни в код, шаблони або конфігурацію, і хочете побачити оновлені зміни на сайті.
- config-edit: Редагує конфігураційні файли Drupal. Ця команда дозволяє вам редагувати файли конфігурації вашого сайту безпосередньо з командного рядка.
- config-export: Експортує конфігурацію Drupal до файлів. Конфігурація Drupal може бути експортована у вигляді файлів, що дозволяє вам зберегти конфігурацію та використовувати її на іншому сайті.
- config-import: Імпортує конфігурацію Drupal з файлів. Ця команда дозволяє вам імпортувати файли конфігурації, які були експортовані з іншого сайту, і застосувати їх до поточного сайту.
- core-cron: Виконує заплановану задачу крона Drupal. Крон - це механізм, який виконує різні фонові задачі на вашому Drupal-сайті, такі як очищення кешу, відправлення листів тощо. Ця команда виконує заплановану задачу крона вручну.
- core-status: Виводить інформацію про статус Drupal-сайту. Ця команда надає загальну інформацію про ваш Drupal-сайт, включаючи версію, режим оточення, інформацію про базу даних тощо.
- entity:delete: Видаляє сутність Drupal, наприклад, вузли, користувачі, терміни таксономії тощо. Ви можете вказати тип сутності та ідентифікатор(и), які потрібно видалити.
- entity:info: Виводить інформацію про сутності Drupal, такі як типи вузлів, поля, таксономія тощо. Ця команда надає список доступних сутностей та їх властивостей.
- help: Виводить довідку для команд Drush або для конкретної команди. Ви можете використовувати цю команду, щоб отримати опис команди, список параметрів та приклади використання.

- module:download: Завантажує модуль Drupal. Ця команда дозволяє вам завантажити модуль з репозиторію проектів Drupal.org та встановити його на вашому сайті.
- module:enable: Активує модуль Drupal. Ви можете вказати ім'я модуля, який потрібно активувати, і Drush здійснить відповідну дію.
- module:install: Встановлює модуль Drupal. Ви можете вказати ім'я модуля, який потрібно встановити, і Drush встановить його залежності та активує.
- module:list: Виводить список встановлених модулів Drupal. Ця команда надає список всіх встановлених модулів на вашому сайті.
- pm-update: Оновлює модулі Drupal до останньої версії. Ця команда перевіряє наявні оновлення модулів та здійснює процес оновлення до останньої доступної версії.
- pm-updatecode: Оновлює код модулів Drupal до останньої версії. Ця команда перевіряє наявні оновлення коду модулів та здійснює процес оновлення до останньої доступної версії.
- sql-cli: Виконує SQL-запити до бази даних Drupal. Ця команда відкриває консольний клієнт бази даних, де ви можете виконувати запити безпосередньо до бази даних вашого Drupal-сайту.
- sql-dump: Створює резервну копію бази даних Drupal. Ця команда створює SQL-дамп вашої бази даних, який містить всю структуру та дані.
- theme:enable: Активує тему Drupal. Ви можете вказати ім'я теми, яку потрібно активувати, і Drush здійснить відповідну дію.
- user-create: Створює нового користувача Drupal. Ця команда дозволяє вам створити новий обліковий запис користувача на вашому Drupal-сайті.
- user-login: Виводить URL-адресу для входу в систему як адміністратор. Ця команда виводить URL-адресу, за допомогою якого ви можете швидко увійти в систему як адміністратор вашого Drupal-сайту.

Це лише декілька основних команд Drush. Drush надає багато інших команд, які допомагають вам керувати та автоматизувати ваш Drupal-сайт. Ви можете докладніше ознайомитися з усіма командами, виконавши команду `drush help` або переглянувши офіційну документацію Drush.

2.2.5. ВСТАНОВЛЕННЯ МОДУЛІВ ТА ТЕМ

Для встановлення модуля в Drupal 8 за допомогою Composer слід виконати наступні кроки:

Відкрийте командний рядок (термінал) та перейдіть до кореневої папки вашого Drupal-проекту.

Виконайте наступну команду для встановлення модуля за допомогою Composer:

composer require drupal/[module_name]

Замініть `[module_name]` на ім'я модуля, який ви хочете встановити. Наприклад, якщо ви хочете встановити модуль "Views", команда буде виглядати так:

composer require drupal/views

Composer почне завантажувати модуль та всі його залежності. Якщо у модуля є залежності, вас можуть попросити підтвердити встановлення цих залежностей. Введіть "y" або "yes" і натисніть Enter, щоб продовжити.

Після завершення встановлення модуля Composer автоматично оновить файл `composer.json` та `composer.lock`, а також скопіює модуль до папки `modules` вашого Drupal-проекту.

Переконайтеся, що ваша система виконує автозавантаження (autoloading) згідно з налаштуваннями Composer. Це забезпечить, що модуль буде коректно завантажений в Drupal.

Увійдіть до адміністративного інтерфейсу Drupal та перейдіть на сторінку "Extend" (Розширення). Там ви побачите встановлений модуль у списку модулів. Встановіть прапорець поруч з модулем та натисніть кнопку "Install" (Встановити) внизу сторінки.

Після цих кроків модуль буде успішно встановлений в вашому Drupal-проекті за допомогою Composer. Composer допомагає керувати залежностями та оновленнями модулів, тому рекомендується використовувати цей підхід для управління модулями в Drupal 8.

Для встановлення модуля в Drupal 8 вручну слід виконати наступні кроки:

Завантажте модуль: Завантажте архів модуля, який ви хочете встановити, з офіційного сайту Drupal або іншого надійного джерела. Зазвичай модуль буде завантажений у вигляді ZIP-архіву.

Розпакуйте архів: Розпакуйте архів модуля і скопіюйте його в папку `modules` вашого Drupal-сайту. Ця папка зазвичай знаходиться за шляхом `sites/all/modules` або `modules/custom`. Якщо папки `modules` не існує, створіть її.

Оновлення автозавантаження: Після розміщення модуля в папку `modules`, вам може знадобитись оновити автозавантаження, щоб Drupal розпізнав новий модуль. Це можна зробити за допомогою адміністративного інтерфейсу Drupal або виконавши команду `drush cr` з командного рядка.

Встановлення модуля: Щоб встановити модуль, увійдіть до адміністративного інтерфейсу Drupal. Перейдіть до сторінки "Extend" (Розширення), де ви побачите список доступних модулів. Знайдіть модуль, який ви хочете встановити, та поставте прапорець поруч з ним. Потім натисніть кнопку "Install" (Встановити) внизу сторінки.

Виконання оновлень: Після встановлення модуля Drupal може запросити виконання оновлень бази даних. Це може бути необхідно, якщо модуль має зміни в базі даних, що потребують оновлення. Виконайте необхідні оновлення, якщо вони зазначені.

Активація модуля: Після встановлення модуль зазвичай необхідно активувати. Перейдіть до сторінки "Extend" (Розширення) в адміністративному інтерфейсі Drupal, знайдіть модуль у списку та поставте прапорець поруч з ним. Потім натисніть кнопку "Install" (Встановити) внизу сторінки.

Після цих кроків модуль буде встановлений і готовий до використання на вашому Drupal-сайті. Будьте уважні при встановленні модулів і переконайтеся, що ви використовуєте модулі, які походять від надійних джерел і сумісні з вашою версією Drupal.

При інсталяції модулів варто розуміти правильну структуру директорій, прийняту в Drupal 8 за стандартну.

Структура директорій в Drupal 8 дещо відрізняється від попередніх версій Drupal. Ось основні директорії, які ви знайдете в Drupal 8:

- core: Це головна директорія, в якій знаходиться ядро Drupal 8. Вона містить всі файлів і папки, пов'язані з ядром системи.

- modules: Ця директорія містить усі модулі Drupal. Кожен модуль знаходиться в окремій піддиректорії зі своїм ім'ям. Модулі використовуються для розширення функціональності Drupal.
- themes: У цій директорії розміщуються всі теми Drupal. Кожна тема знаходиться в окремій піддиректорії зі своїм ім'ям. Теми використовуються для зовнішнього вигляду та стилізації вашого Drupal-сайту.
- profiles: Якщо ви використовуєте встановлення Drupal з використанням дистрибутиву або інсталяційного профілю, ця директорія містить профілі інсталяції. Кожен профіль знаходиться в окремій піддиректорії.
- libraries: У цій директорії можуть знаходитися зовнішні бібліотеки, які використовуються вашим Drupal-сайтом.
- sites: Ця директорія містить всі файли, пов'язані з конфігурацією та налаштуванням вашого сайту. Вона містить піддиректорії для кожного сайту або області сайту (якщо використовується мультисайтова конфігурація).
- sites/default: Цей піддиректорій містить конфігураційні файли та налаштування для типового сайту.
- sites/example.com: Цей піддиректорій може містити конфігураційні файли та налаштування для окремого сайту, якщо ви використовуєте мультисайтову кофігурацію.
- vendor: У цій директорії розміщуються залежності, встановлені за допомогою Composer. Вона містить всі залежності PHP-пакетів, які використовуються в Drupal.
- web: Це папка, що визначає веб-корінь вашого Drupal-сайту. Вона містить всі публічно доступні файли, такі як індексний файл, зображення, CSS-файли тощо.
- config: У цій директорії зберігаються файли конфігурації вашого Drupal-сайту. Вона містить налаштування, які ви зберігаєте через адміністративний інтерфейс Drupal.
- tmp: Ця директорія використовується для тимчасових файлів, які створюються під час роботи Drupal, таких як завантажені файли або тимчасові файли кешу.

- Це основні директорії в Drupal 8, але структура може змінюватися в залежності від вашого проекту та налаштувань.

Для встановлення теми в Drupal 8 за допомогою Composer слід виконати наступні кроки:

Відкрийте командний рядок (термінал) та перейдіть до кореневої папки вашого Drupal-проекту.

Виконайте наступну команду для встановлення теми за допомогою Composer:

composer require drupal/[theme_name]

Замініть `[theme_name]` на ім'я теми, яку ви хочете встановити. Наприклад, якщо ви хочете встановити тему "Bootstrap", команда буде виглядати так:

composer require drupal/bootstrap

Composer почне завантажувати тему та всі її залежності. Якщо у теми є залежності, вас можуть попросити підтвердити встановлення цих залежностей. Введіть "y" або "yes" і натисніть Enter, щоб продовжити.

Після завершення встановлення теми Composer автоматично оновить файл `composer.json` та `composer.lock`, а також скопіює тему до папки `themes` вашого Drupal-проекту.

Переконайтеся, що ваша система виконує автозавантаження (autoloading) згідно з налаштуваннями Composer. Це забезпечить, що тема буде коректно завантажена в Drupal.

Увійдіть до адміністративного інтерфейсу Drupal та перейдіть на сторінку "Appearance" (Вигляд). Там ви побачите встановлену тему у списку тем. Встановіть прапорець поруч з темою та натисніть кнопку "Set default" (Встановити за замовчуванням) або "Install and set as default" (Встановити і встановити за замовчуванням) внизу сторінки.

Після цих кроків тема буде успішно встановлена в вашому Drupal-проекті за допомогою Composer. Composer допомагає керувати залежностями та оновленнями тем, тому рекомендується використовувати цей підхід для управління темами в Drupal.

При встановленні модулів або тем варто пам’ятати про правила керування версіями в composer, які дозволяють забезпечити оптимальні політики роботи з версіями.

Керування версіями в Composer дозволяє точно визначити, яку версію пакету ви хочете встановити або які обмеження версій мають бути застосовані до пакетів. Основні поняття, які використовуються для керування версіями в Composer, включають наступне:

Точна версія: Ви можете вказати точну версію пакету, яку хочете встановити. Наприклад:

composer require vendor/package:1.2.3

Це означає, що Composer встановить саме версію 1.2.3 цього пакету.

Діапазон версій: Ви можете використовувати діапазон версій для обмеження версій пакету. Наприклад:

composer require vendor/package:~1.2

Це означає, що Composer встановить найновішу версію пакету, яка належить до версій 1.2.x (де x - будь-яка версія).

Оператори порівняння: Ви можете використовувати оператори порівняння для встановлення більш детальних обмежень версій. Деякі з операторів порівняння включають:

- `<`: Менше ніж.
- `<=`: Менше або дорівнює.
- `>`: Більше ніж.
- `>=`: Більше або дорівнює.
- `!=`: Не дорівнює.
- `~`: Часткова сумісність. Наприклад, `~1.2` встановить будь-яку версію пакету, яка належить до версій 1.2.x, включаючи найновішу стабільну версію.

Застарілі версії: Ви також можете вказати певні версії як застарілі, щоб Composer не встановлював їх. Наприклад:

composer require vendor/package:^1.0@dev

Це означає, що Composer не буде встановлювати будь-яку версію пакету, яка належить до версій 1.x (де x - будь-яка версія), якщо вона є застарілою.

Конфлікти та залежності: Крім вказання версій для окремих пакетів, Composer також автоматично вирішує залежності між пакетами та може генерувати конфлікти, якщо виникають проблеми з сумісністю версій між пакетами.

Файл composer.lock: Composer створює файл `composer.lock`, в якому зберігається точна інформація про встановлені версії пакетів та їх залежностей. Це дозволяє забезпечити репродукованість встановлених версій пакетів на різних середовищах розробки.

Керування версіями в Composer дозволяє вам гнучко керувати версіями пакетів у вашому проекті та забезпечує сумісність між залежностями. Це важливий інструмент для ефективного управління залежностями та розгортанням веб-проектів.

2.2.6. БАЗОВІ НАЛАШТУВАННЯ

Після успішної інсталяції Drupal 8 вам слід виконати деякі базові налаштування для вашого проекту. Ось кілька кроків, які рекомендується виконати:

Запустіть налаштування сайту: Після інсталяції відкрийте свій веб-переглядач та перейдіть до домашньої сторінки вашого Drupal-сайту. Вас буде вітати сторінка налаштування сайту. Слідуйте інструкціям на екрані для налаштування мови, бази даних, адміністратора та інших основних параметрів сайту.

Перевірте налаштування файлів: Переконайтеся, що налаштування файлів вашого Drupal-сайту вірні. Відкрийте файл `settings.php`, який знаходиться в папці `sites/default`, та переконайтеся, що налаштування для бази даних та інші налаштування вірні.

Встановіть зовнішній вигляд: За замовчуванням Drupal 8 поставляється з базовою темою "Seven". Однак, ви можете встановити іншу тему, яка краще підходить для вашого проекту. Перейдіть до адміністративного інтерфейсу Drupal і перейдіть на сторінку "Appearance" (Вигляд), де ви зможете вибрати тему для свого сайту та налаштувати її параметри.

Активуйте та налаштуйте модулі: Drupal 8 поставляється зі значною кількістю модулів, які ви можете активувати та налаштувати відповідно до потреб вашого проекту. Перейдіть до сторінки "Extend" (Розширення) в адміністративному інтерфейсі Drupal, де ви зможете переглянути список доступних модулів та активувати ті, які вам потрібні.

Перевірте налаштування безпеки: Drupal 8 має деякі вбудовані налаштування безпеки, які рекомендується перевірити і налаштувати згідно з вашими потребами. Зверніть увагу на такі аспекти безпеки, як доступ до адміністративного інтерфейсу, права доступу користувачів, паролі та захист від зламу.

Створіть основний контент: Залежно від призначення вашого сайту, створіть необхідний основний контент, такий як сторінки, статті, меню та інше. Це допоможе вам перевірити правильність відображення та функціональності вашого Drupal-сайту.

Оновіть конфігурацію кешу: Після налаштування вашого Drupal-сайту рекомендується оновити конфігурацію кешу. Ви можете це зробити у розділі "Performance" (Продуктивність) на сторінці "Configuration" (Конфігурація) в адміністративному інтерфейсі. Оновлення кешу може покращити продуктивність та швидкодію вашого сайту.

Це лише декілька базових налаштувань, які ви можете виконати після інсталяції Drupal 8. Залежно від вашого проекту можуть бути потрібні додаткові кроки налаштування.

Drupal 8 поставляється зі значною кількістю модулів, які ви можете використовувати для розширення функціональності вашого сайту. Ось кілька рекомендованих модулів, які можуть бути корисними в багатьох проектах на Drupal 8:

Views: Цей модуль дозволяє вам створювати потужні запити до бази даних та налаштовувати вигляди для відображення контенту на вашому сайті. Він надає гнучкий інтерфейс для створення списків, таблиць, фільтрів та інших компонентів веб-інтерфейсу.

Pathauto: Цей модуль автоматично створює людяні URL-адреси для сторінок контенту на вашому сайті, замість стандартних URL-адресів, що створюються Drupal. Він дозволяє вам налаштовувати шаблони URL-адресів залежно від типу контенту та його поля.

Token: Цей модуль розширює можливості Drupal для використання замінників (токенів) у текстових полях, налаштуваннях шаблонів та інших місцях. Він дозволяє

вам вставляти динамічні значення, такі як назви користувачів, назви сайту, дати тощо, за допомогою спеціальних токенів.

Path Redirect: Цей модуль дозволяє створювати перенаправлення зі старих URL-адресів на нові. Він корисний, якщо ви змінили структуру URL-адресів або перенесли контент з іншого сайту.

Metatag: Цей модуль дозволяє вам налаштовувати метатеги, такі як заголовок сторінки, опис та ключові слова, для кожної сторінки вашого сайту. Це допомагає покращити пошукову оптимізацію (SEO) вашого контенту.

CKEditor: Цей модуль замінює стандартний текстовий редактор Drupal на потужний редактор CKEditor. Він надає багато функціональних можливостей для форматування тексту та додавання медіа-елементів.

Webform: Цей модуль дозволяє вам створювати різні типи форм на вашому сайті. Він має інтуїтивний інтерфейс для створення форм та збору даних від користувачів.

Paragraphs: Цей модуль дозволяє вам створювати складні структури контенту шляхом комбінування різних типів параграфів. Він дозволяє вам створювати повторювані групи полів або компонентів контенту, що дозволяє більш гнучко організовувати ваш контент.

Media: Цей модуль надає підтримку для управління медіа-файлами, такими як зображення, відео та аудіо, на вашому сайті. Він дозволяє вам завантажувати, організовувати та вставляти медіа-елементи у ваш контент.

Це лише кілька прикладів модулів, які можуть бути корисними в Drupal 8. Вибір модулів залежить від потреб вашого проекту та функціональності, яку ви хочете реалізувати на своєму сайті.

Для ведення розробки варто здійснити ряд операцій, які значно спрощують процес. Щоб відключити кешування та включити режим налагодження Twig (Twig Debug) в Drupal 8, слід виконати наступні кроки:

Відкрийте файл `sites/default/settings.php` вашого Drupal-сайту у текстовому редакторі.

Знайдіть наступний рядок коду:

```
$settings['cache']['bins']['render'] = 'cache.backend.null';
```

Якщо такого рядка немає, вставте його у файл.

Закоментуйте цей рядок, додавши `//` в початок рядка:

```
// $settings['cache']['bins']['render'] = 'cache.backend.null';
```

Додайте наступні рядки коду після рядка, який ви коментували:

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

// Enable Twig debugging.
$config['twig.config']['debug'] = TRUE;
$config['twig.config']['auto_reload'] = TRUE;
$config['twig.config']['cache'] = FALSE;
```

Збережіть зміни у файлі `settings.php`.

Зайдіть до адміністративного інтерфейсу Drupal та перейдіть на сторінку "Configuration" (Конфігурація) -> "Development" (Розробка) -> "Performance" (Продуктивність). Виберіть пункт "Clear all caches" (Очистити всі кеші) та натисніть кнопку "Clear caches" (Очистити кеші).

Після цього кешування буде вимкнено, а режим налагодження Twig буде включено. Ви побачите шаблони Twig у розкодованому форматі та зможете отримати більш детальну інформацію про шаблони та змінні, включаючи рядки та імена файлів шаблонів.

Ці зміни дозволять вам розробляти та відлагоджувати Drupal-сайт, необхідний для відключення кешування та включення режиму налагодження Twig. Не забудьте повернути налаштування до значень за замовчуванням на продуктивному сервері.

Питання для самоконтролю:

1. Які розгорнути проект?
2. Як інсталювати Drupal з використанням Composer та в ручному режимі?
3. Як використовувати Drush?
4. Які операції потрібно провести після інсталяції?

2.3. СУТНОСТІ (ENTITIES) В DRUPAL

Сутності (entities) в Drupal 8 є однією з ключових нововведень у порівнянні з попередніми версіями Drupal. Вони пропонують більш структурований та розширюваний підхід до зберігання та управління контентом на вашому сайті. Ось деякі важливі аспекти сутностей в Drupal 8:

Концепція сутностей: Сутності представляють об'єкти або елементи даних, які ви хочете зберігати та управляти в своєму Drupal-сайті. Вони можуть бути різних типів, таких як сторінки, статті, користувачі, коментарі тощо. Кожна сутність має свої властивості, які визначаються полями.

Сутність як тип даних: Сутності в Drupal 8 вважаються типом даних, а не просто записом у базі даних. Вони мають типову структуру, включаючи поля з визначеними типами даних, відносини між сутностями та додаткові метадані.

Сутність як розширюваний об'єкт: Сутності можуть бути розширені за допомогою модулів, які надають додаткові поля та функціональність. Ви можете використовувати модулі для розширення вбудованих сутностей або створення власних типів сутностей.

Управління властивостями: Кожна сутність має свої властивості, які визначаються полями. Поля можуть бути різних типів, таких як текст, число, дата, зображення тощо. Ви можете налаштовувати поля сутностей, визначати обов'язковість, валідацію та інші правила.

Система подій і хуків: Drupal 8 надає систему подій та хуків, які дозволяють реагувати на дії, пов'язані з сутностями. Це дозволяє вам в пливати на збереження, видалення, оновлення сутностей та виконувати додаткові дії відповідно.

Мовна підтримка: Сутності в Drupal 8 мають вбудовану підтримку багатомовності. Ви можете зберігати та відображати контент у різних мовах, налаштовувати мовні варіанти для полів сутностей та контролювати мову відображення контенту.

Сутності в Drupal 8 надають більш гнучкий, розширюваний та структурований підхід до зберігання та управління контентом на вашому сайті. Вони дозволяють більш ефективно організувати ваші дані та розширювати функціональність за допомогою модулів.

2.3.1. ПОНЯТТЯ СУТНОСТІ. ОСНОВНІ СУТНОСТІ В DRUPAL

У Drupal 8 сутності (entities) є основною концепцією для представлення та управління даними. Сутності використовуються для зберігання, обробки та відображення різноманітного контенту на вашому сайті. Вони можуть представляти такі об'єкти, як статті, сторінки, користувачі, зображення та багато іншого.

Основні особливості сутностей в Drupal 8 включають:

Структурованість: Сутності в Drupal 8 мають чітку структуру, описану у вигляді полів. Кожне поле визначає тип даних та властивості, такі як заголовок, текст, дата, зображення тощо.

Система поліморфізму: Сутності можуть мати різні типи та розширення. Наприклад, ви можете мати сутність "Стаття" та сутність "Сторінка", які мають різні поля та функціональність.

Модульність: Сутності можуть бути розширені та налаштовані за допомогою модулів. Ви можете додавати свої власні поля до сутностей, створювати нові типи сутностей та визначати взаємодію між ними.

Дозволи доступу: Сутності використовують систему дозволів Drupal для керування доступом користувачів до контенту. Ви можете встановлювати права доступу до окремих сутностей, полями та функціями на основі ролей користувачів.

Мовна підтримка: Сутності в Drupal 8 мають вбудовану підтримку багатомовності. Ви можете зберігати та відображати контент у різних мовах, а також налаштовувати мовні варіанти для полів сутностей.

Сутності в Drupal 8 дозволяють розширювати функціональність вашого сайту та створювати різноманітний контент згідно з вашими потребами. Ви можете створювати власні типи сутностей, визначати поля, реляції між сутностями та використовувати розширені можливості Drupal API для маніпуляції даними.

У Drupal 8 існує кілька видів сутностей, які можна використовувати для зберігання та управління різноманітним контентом на вашому сайті. Основні види сутностей в Drupal 8 включають:

Вбудовані сутності (Built-in entities): Drupal 8 поставляється з кількома вбудованими типами сутностей, такими як статті (node), коментарі (comment), користувачі (user), події календаря (event), зображення (image) та інші. Вони мають базові поля, такі як заголовок, вміст, дата публікації тощо.

Таксономія (Taxonomy): Таксономія використовується для організації та класифікації контенту на вашому сайті. Ви можете створювати терміни та категорії, які використовуються для тегування та групування сутностей. Таксономія може бути використана для створення структурованих наборів термінів, таких як категорії, теги, розділи тощо.

Форми контактів (Contact forms): Drupal 8 має вбудований модуль "Контакт", який дозволяє вам створювати форми контактів. Ви можете налаштовувати поля форми, призначати одержувачів повідомлень та використовувати їх для отримання зворотного зв'язку від відвідувачів сайту.

Користувацькі сутності (Custom entities): Drupal 8 надає вам можливість створювати власні типи сутностей. Ви можете визначати свої поля, реляції, додавати власну бізнес-логіку та розширювати функціональність сутностей за допомогою модулів.

Медіа (Media): У Drupal 8 існує поняття медіа-сутностей, які дозволяють вам керувати зображеннями, відео, аудіо та іншими медіа-файлами. Ви можете створювати колекції медіа, встановлювати атрибути, такі як опис, автор та джерело, та використовувати їх у своєму контенті.

Це лише кілька прикладів видів сутностей, які можна використовувати в Drupal 8. Залежно від потреб вашого проекту, ви можете створювати власні типи сутностей та використовувати розширені можливості для управління контентом.

Використання сутностей в Drupal 8 надає кілька переваг, які допомагають управляти та організовувати контент на вашому сайті. Ось кілька переваг використання сутностей в Drupal 8:

Гнучкість та розширюваність: Сутності в Drupal 8 дозволяють вам створювати власні типи сутностей з необхідними полями та властивостями. Ви можете легко налаштовувати сутності згідно з потребами вашого проекту та розширювати їх функціональність за допомогою модулів.

Модульність: Сутності використовують модульну архітектуру Drupal, що дозволяє створювати незалежні модулі для кожного типу сутності. Це спрощує управління та підтримку коду, а також забезпечує гнучкість при розробці та налаштуванні сутностей.

Повторне використання та спільне використання: Сутності можуть бути використані в різних частинах вашого сайту, дозволяючи повторно використовувати

контент та компоненти. Наприклад, ви можете мати сутність "Банер", яку використовуєте на домашній сторінці, сторінках проектів та інших місцях.

Підтримка багатомовності: Сутності в Drupal 8 мають вбудовану підтримку багатомовності. Ви можете зберігати та відображати контент у різних мовах, а також налаштовувати мовні варіанти для полів сутностей.

Пошукова оптимізація: Сутності в Drupal 8 надають можливості для налаштування метатегів, URL-адрес та інших аспектів, що сприяють пошуковій оптимізації (SEO). Ви можете легко налаштовувати метатеги для сторінок, поля для пошукової індексації та інші параметри для поліпшення видимості вашого контенту в пошукових системах.

Управління дозволами: Сутності використовують систему дозволів Drupal для керування доступом користувачів до контенту. Ви можете встановлювати права доступу до окремих сутностей, полів та функцій на основі ролей користувачів.

Використання сутностей в Drupal 8 дозволяє вам ефективно управляти та розширювати контент вашого сайту. Вони надають гнучкість, розширюваність та можливості для налаштування, що робить Drupal 8 потужною платформою для створення різноманітних веб-проектів.

2.3.2. ТИПИ КОНТЕНТУ, ПОЛЯ, НАЛАШТУВАННЯ ФОРМ І РЕЖИМІВ ВІДОБРАЖЕННЯ. НОДИ.

Типи контенту в Drupal 8 є одним із способів організації та категоризації різних видів контенту на вашому сайті. Вони використовуються для визначення структури та полів, які будуть використовуватися при створенні нового контенту. Ось кілька важливих аспектів типів контенту в Drupal 8:

Визначення структури: Типи контенту дозволяють вам визначити структуру для конкретного виду контенту. Ви можете встановлювати поля, властивості та поведінку для цього типу контенту. Наприклад, ви можете мати тип контенту "Стаття" з полями, такими як заголовок, вміст, автор, дата публікації тощо.

Поля та типи полів: Ви можете використовувати різні типи полів, такі як текстові, числові, вибір зі списку, зображення, дата тощо, для визначення даних, які будуть вводитися при створенні контенту. Крім того, ви можете налаштовувати вимоги до полів, валідацію та інші параметри.

Розширення типів контенту: Ви можете розширити типи контенту за допомогою модулів. Це дозволяє вам додавати додаткові поля, властивості та функціональність до

існуючих типів контенту. Наприклад, ви можете додати поле зображення до типу контенту "Стаття", щоб дозволити користувачам додавати зображення до своїх статей.

Дозволи доступу: Drupal 8 надає гнучку систему керування дозволами для типів контенту. Ви можете встановлювати права доступу для створення, редагування та перегляду контенту для різних ролей користувачів. Це дозволяє вам контролювати, хто може створювати та редагувати різні типи контенту на вашому сайті.

Шаблони відображення: Для кожного типу контенту в Drupal 8 можна налаштовувати окремі шаблони відображення. Це дозволяє вам контролювати, як саме контент цього типу буде відображатися на вашому сайті, включаючи порядок та стиль полів, використання розмітки тощо.

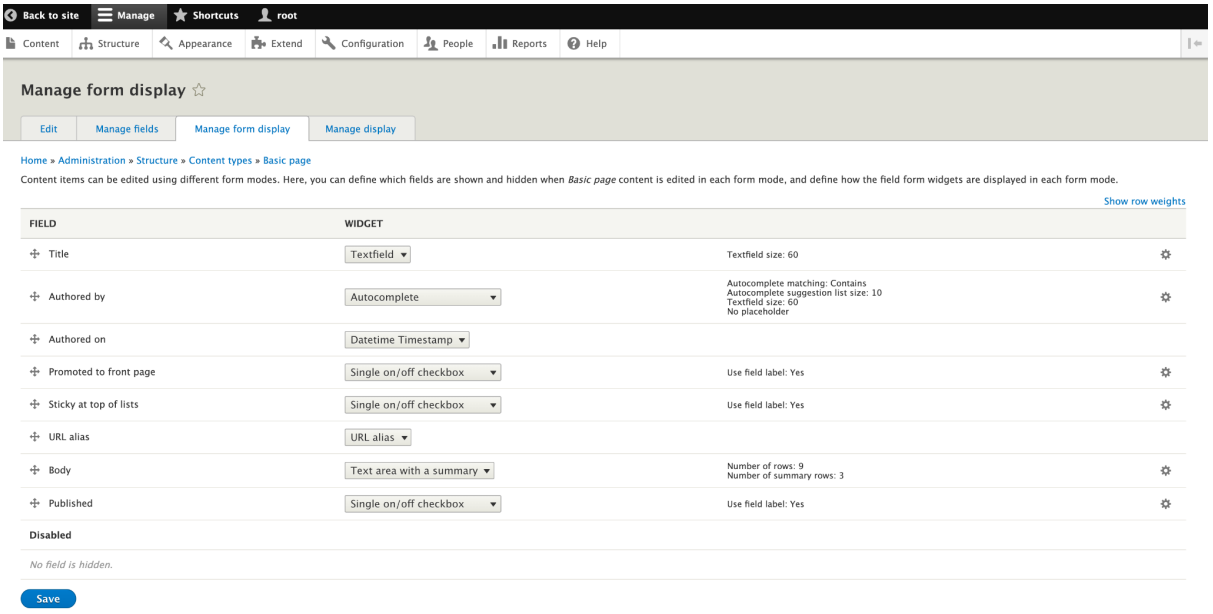

Рис. 3. Сторінка налаштування відображення.

Мовна підтримка: Типи контенту в Drupal 8 підтримують багатомовність. Ви можете налаштовувати мовні варіанти для полів та контролювати, який контент буде відображатися для різних мов.

Використання типів контенту дозволяє вам більш гнучко керувати структурою та властивостями контенту на вашому Drupal-сайті. Ви можете визначати власні типи контенту, розширювати їх функціональність та налаштовувати відображення контенту залежно від вашої потреби.

В Drupal 8 існує кілька типів полів, які можна використовувати для створення різних типів даних в вашому сайті. Ось опис деяких типів полів, доступних в Drupal 8:

- Text (Текст): Тип полів "Текст" дозволяє вводити короткі текстові значення, такі як заголовки, імена, описи тощо.
- Long text (Довгий текст): Поле "Довгий текст" призначене для вводу більш довгих текстових значень, таких як вміст статті, описи, коментарі тощо. Ви можете використовувати його для форматованого тексту, включаючи HTML-код та форматування.
- Integer (Ціле число): Поле "Ціле число" дозволяє вводити цілі числа без десяткової частини, такі як кількість елементів, порядкові номери тощо.
- Decimal (Десяткове число): Поле "Десяткове число" дозволяє вводити числа з десятковою точкою, такі як ціни, вага тощо. Воно може містити як цілу, так і десяткову частину.
- Boolean (Булеве значення): Поле "Булеве значення" дозволяє вибирати між двома варіантами: "так" або "ні". Воно використовується для зберігання логічних значень, таких як стан активації, прапори тощо.
- Date (Дата): Тип полів "Дата" дозволяє вводити дату або діапазон дат. Ви можете використовувати його для встановлення дати публікації, дати народження тощо.
- Entity reference (Посилання на сутність): Поле "Посилання на сутність" дозволяє вибирати сутність з існуючих сутностей Drupal, таких як статті, користувачі, терміни таксономії тощо. Воно використовується для встановлення взаємозв'язків між різними типами контенту.
- File (Файл): Тип полів "Файл" дозволяє завантажувати та зберігати файли, такі як зображення, відео, документи тощо. Ви можете обмежувати типи файлів, встановлювати максимальний розмір тощо.
- Image (Зображення): Поле "Зображення" спеціалізоване для завантаження та зберігання зображень. Воно має додаткові можливості, такі як автоматична генерація мініатюр, масштабування зображень тощо.
- Term reference (Посилання на термін таксономії): Поле "Посилання на термін таксономії" дозволяє вибирати терміни зі структури таксономії. Воно використовується для категоризації та організації контенту.

Це лише деякі типи полів, доступні в Drupal 8. Ви також можете використовувати розширені типи полів, що надаються модулями, або створювати власні типи полів для вирішення специфічних потреб вашого проекту.

Проектування типів контенту в Drupal 8 включає кроки, які допоможуть вам визначити структуру та властивості контенту, необхідного для вашого сайту. Ось деякі кроки, які можна виконати при проектуванні типу контенту в Drupal 8:

Аналіз вимог: Почніть з аналізу вимог вашого проекту. Зрозумійте, який контент вам потрібен, які дані повинні бути збережені та які функціональність необхідна. Розгляньте різні аспекти контенту, такі як заголовки, вміст, зображення, категорії, теги тощо.

Визначення полів: Визначте необхідні поля для вашого типу контенту. Розгляньте, які типи даних вам потрібні для кожного поля (текст, число, дата тощо) та які обов'язкові поля вам потрібні.

Встановлення відносин: Розгляньте, чи потрібно встановлювати відносини між вашим типом контенту та іншими сутностями в Drupal, такими як користувачі, статті, категорії тощо. Наприклад, можливо, ви хочете мати поле "Автор", яке посилається на користувача, що створив контент.

Встановлення дозволів доступу: Розгляньте, які права доступу до вашого типу контенту повинні мати різні ролі користувачів на вашому сайті. Встановіть права доступу до створення, редагування та перегляду контенту для кожної ролі.

Конфігурація відображення: Визначте, як ваш контент буде відображатися на сайті. Розгляньте використання шаблонів, налаштування форматування, розмітки та стилів для відображення полів контенту.

Розширення та модулі: Розгляньте, чи необхідно розширювати ваш тип контенту за допомогою модулів. В Drupal 8 існує багато модулів, які надають додаткові функціональність, яку можна використовувати для вашого типу контенту.

Тестування і налагодження: Після налаштування типу контенту виконайте тестування та налагодження, щоб переконатися, що все працює як очікувалося. Перевірте, чи дані зберігаються правильно, чи відображаються на сайті та чи працюють налаштування дозволів доступу.

Проектування типів контенту в Drupal 8 є процесом, який вимагає аналізу, визначення потреб і налаштування параметрів для відповідності вашим вимогам. Це дозволяє вам створити структурований та гнучкий тип контенту, який задовольняє потреби вашого проекту.

Рішення про повторне використання полів в сутностях Drupal 8 залежить від конкретних вимог та ситуації проекту. Ось кілька розглядуваних випадків, коли варто або не варто повторно використовувати поля в сутностях:

Коли варто повторно використовувати поля:

- Однотипні дані: Якщо у вас є сутності, які мають однаковий тип даних та структуру, то варто розглянути можливість повторного використання полів. Наприклад, якщо у вас є дві сутності "Стаття" та "Блог", які мають однакову структуру полів, то можна створити один тип поля "Текст" та використовувати його в обох сутностях.
- Збереження часу та зусиль: Повторне використання полів дозволяє економити час та зусилля при створенні та керуванні сутностями. Ви можете налаштувати поле один раз і використовувати його у багатьох сутностях, замість повторного налаштування для кожної сутності окремо.

Коли не варто повторно використовувати поля:

- Різні типи даних: Якщо ваші сутності мають різні типи даних або структуру, то не рекомендується повторне використання полів. Наприклад, якщо у вас є сутність "Стаття" з полем "Текст" та сутність "Користувач" з полем "Електронна пошта", то краще мати окремі поля для кожної сутності.
- Різні правила валідації або форматування: Якщо ваші сутності вимагають різні правила валідації або форматування для полів, то краще мати окремі поля для кожної сутності. Наприклад, якщо у вас є сутність "Стаття" з полем "Текст" та сутність "Користувач" з полем "Електронна пошта", то краще мати окремі поля для кожної сутності.

Загалом, варто розглядати повторне використання полів у сутностях, коли це логічно та виправдано з точки зору структури даних та зручності управління. Проте, варто ретельно аналізувати конкретні вимоги проекту та зважати на можливі наслідки, щоб прийняти найкраще рішення для вашого проекту.

Налаштування форми редагування контенту типу в Drupal 8 включає кілька кроків, які дозволяють вам змінювати вигляд та функціональність форми редагування. Ось декілька шляхів для налаштування форм редагування контенту:

Використання Display Suite або Panels: Модулі Display Suite та Panels дозволяють вам використовувати інтерфейс з перетягуванням для налаштування вигляду форм редагування. Ви можете вибирати, які поля відображати, в якому порядку та в якій розмітці.

Використання Manage Display: У адміністративному розділі "Structure" -> "Content types" -> "Manage display" ви можете вибрати, які поля відображати і в якому форматі на формі редагування. Ви можете налаштовувати заголовки полів, форматування тексту, використовувати поліпшення форматування через модуль "Field Group" та інші параметри.

Форми виправлення полів: Ви можете використовувати модуль "Field Group" для групування полів та створення колапсованих секцій на формі редагування. Це дозволяє покращити організацію полів та спростити форму.

Додаткові модулі: У Drupal 8 існує багато додаткових модулів, які дозволяють налаштовувати форми редагування контенту. Наприклад, модуль "Field Permissions" дозволяє налаштовувати права доступу до полів, модуль "Conditional Fields" дозволяє відображати поля в залежності від значень інших полів, модуль "Field Group" дозволяє групувати поля та створювати колапсовані секції тощо. Розгляньте можливості цих модулів для налаштування форм редагування.

Розробка шаблонів: Ви можете розробляти власні шаблони для форм редагування контенту, використовуючи тему вашого сайту. Шаблони можуть дати вам більшу гнучкість та контроль над виглядом форми редагування, дозволяючи вам змінювати HTML-структуру та стилізацію.

Враховуйте, що вищенаведені підходи можуть вимагати деякого рівня знань та розуміння Drupal-розробки. Залежно від вашого досвіду та потреб, ви можете вибрати підходи, які найбільше підходять для вашого проекту.

У Drupal 8, для налаштування View Mode (режиму відображення) в типах контенту, вам потрібно виконати наступні кроки:

Увійдіть в адміністративну частину свого сайту Drupal 8 і перейдіть до сторінки "Structure" (Структура).

Оберіть "Content types" (Типи контенту) та виберіть тип контенту, для якого ви хочете налаштувати View Mode.

На сторінці налаштування типу контенту знайдіть вкладку "Manage display" (Керування відображенням) і перейдіть на неї.

У верхній частині сторінки ви побачите перелік доступних View Modes (режимів відображення), таких як "Default" (За замовчуванням), "Teaser" (Зведений опис), "Full content" (Повний вміст) та інші, залежно від налаштувань вашого сайту.

Оберіть View Mode, який ви хочете налаштувати, і відредагуйте налаштування полів для цього режиму відображення. Ви можете змінювати порядок полів, приховувати чи відображати їх, налаштовувати форматування, використовувати поліпшення форматування тощо.

Збережіть зміни, коли ви завершили налаштування View Mode.

Після збереження налаштувань, переконайтеся, що ви використовуєте відповідний View Mode в шаблонах або переглядах (Views), щоб відображати контент у відповідному режимі.

Ці кроки дозволять вам налаштувати, як контент конкретного типу відображатиметься у різних режимах на вашому сайті Drupal 8.

2.3.3. ТАКСОНОМІЯ ТА ЇЇ ВИКОРИСТАННЯ

Таксономія в Drupal 8 є системою категоризації та організації контенту. Вона дозволяє створювати і управляти наборами термінів, які можна призначати контенту для подальшого сортування, фільтрації та навігації.

Основні поняття, пов'язані з таксономією в Drupal 8, включають наступне:

Таксономічний термін: Це елемент таксономії, який представляє собою певну категорію або мітку. Таксономічні терміни можуть бути організовані в ієрархічну структуру (таксономічні терміни можуть мати батьківські та дочірні терміни).

Таксономізація Це процес створення та організації таксономічних термінів. Ви можете створювати терміни та налаштовувати їх характеристики, такі як назва, опис, URL-шлях тощо.

Таксономічні поля: Ви можете створювати таксономічні поля та призначати їх для типів контенту. Це дозволяє прив'язувати контент до відповідних таксономічних термінів. Наприклад, ви можете мати таксономічне поле "Категорія" для статей, де можна вибрати відповідну категорію для кожної статті.

Таксономічні сторінки: Drupal 8 автоматично створює сторінки для кожного таксономічного терміну, де відображається список контенту, пов'язаного з цим терміном.

Таксономічний зв’язок: Це можливість прив'язувати таксономічні терміни до інших сутностей Drupal, таких як вузли, щоб створювати зв'язки та ієрархії між контентом.

Таксономія дозволяє організувати контент на сайті, забезпечує зручну навігацію для користувачів та допомагає впорядкувати контент за різними категоріями. Ви

можете використовувати таксономію для створення структури навігації, фільтрації контенту, пошуку та іншого.

Використання таксономії в Drupal 8 має свої переваги та недоліки, які варто враховувати при розгляді цього функціоналу. Ось кілька переваг та недоліків використання таксономії в Drupal 8:

Переваги використання таксономії в Drupal 8:

Гнучкість організації контенту: Таксономія дозволяє вам створювати ієрархічні структури категорій та міток, що дозволяє гнучко організувати ваш контент. Ви можете створювати різні рівні категорій та встановлювати зв'язки між ними, що дозволяє створювати складні структури контенту.

Покращена навігація та фільтрація: За допомогою таксономії, ви можете легко фільтрувати та навігувати контент за категоріями та мітками. Користувачі можуть швидко знаходити та переглядати контент, пов'язаний з певними термінами таксономії, що полегшує їм орієнтацію на сайті.

Покращена пошукова оптимізація: Використання таксономії може покращити пошукову оптимізацію вашого сайту. Ви можете створювати ключові слова, теги та інші мітки, які допомагають пошуковим системам краще індексувати та категоризувати ваш контент, підвищуючи його видимість у пошукових результатах.

Перевикористання та модульність: Таксономія є розширюваною і може використовуватись в різних сутностях Drupal, таких як вузли, блоги, сторінки тощо. Це дозволяє використовувати одну і ту саму таксономію в різних частинах сайту, що спрощує керування та утримання

Недоліки використання таксономії в Drupal 8:

Обмеження структури: Drupal 8 надає певну структуру для таксономії, але вона може бути обмеженою для деяких сценаріїв. Якщо ви потребуєте складну структуру або більше гнучкості, вам можуть знадобитися додаткові модулі або налаштування.

Синхронізація термінів: Коли використовується таксономія в розподіленому середовищі або на декількох сайтах, синхронізація термінів може бути складною. Необхідно враховувати це при плануванні та управлінні таксономією.

Складність управління: При великій кількості термінів таксономії, керування та підтримка може бути складним. Ретельне планування, назви термінів та правильне використання ієрархії допоможуть уникнути цього.

Вимоги до розробників: Налаштування таксономії може вимагати деякого рівня знань Drupal, зокрема розуміння сутностей, полів та модулів, пов'язаних з таксономією.

Розробники повинні мати досвід роботи з цими аспектами Drupal для ефективного використання таксономії.

Загалом, таксономія в Drupal 8 є потужним інструментом для організації та категоризації контенту. Враховуючи переваги та недоліки, ви можете визначити, чи відповідає вона вашим потребам та вимогам проекту.

Проектування таксономії в Drupal 8 вимагає ретельного планування та уваги до деталей. Ось кілька кроків та рекомендацій для правильного проектування таксономії:

Аналіз потреб: Спочатку визначте, які категорії, мітки або теги ви хочете використовувати для організації вашого контенту. Ретельно проаналізуйте ваш контент та визначте основні теми, категорії або характеристики, які будуть використовуватися для класифікації контенту.

Структура таксономії: Розробіть ієрархічну структуру для вашої таксономії, яка відображає ваші категорії та підкатегорії. Встановіть відношення "батько-дитина" між таксономічними термінами, якщо це потрібно. Ретельне планування структури допоможе забезпечити логічну та зручну організацію контенту.

Назви термінів: Використовуйте зрозумілі та описові назви для термінів таксономії. Вони повинні якомога точніше відображати зміст або характеристики контенту, який буде категоризований за допомогою цих термінів.

Уникайте зайвої глибини: Намагайтеся не перевантажувати таксономію занадто глибокими рівнями вкладеності. Якщо ваша таксономія стає занадто складною або незрозумілою, це може ускладнити управління та використання.

Переходіть від загального до конкретного: Починайте з більш загальних категорій або міток і розширюйте їх до більш конкретних деталей. Це допоможе забезпечити зручну навігацію та пошук для користувачів.

Розгляньте майбутні потреби: При проектуванні таксономії враховуйте потенційні майбутні потреби. Запитайте себе, чи плануєте ви розширювати ваш контент у майбутньому, і які категорії або мітки можуть знадобитись вам в подальшому. Вибір гнучкої та розширюваної структури таксономії може полегшити майбутні зміни та розширення.

Розгляньте використання модулів: В Drupal 8 існує багато модулів, які розширюють функціонал таксономії. Розгляньте можливості модулів, таких як Taxonomy Manager, Entity Reference, Term Reference Tree та інші, які можуть полегшити управління та використання таксономії.

Правильне проектування таксономії допоможе вам ефективно організувати та категоризувати контент на вашому сайті. Зверніть увагу на потреби вашого проекту та специфіку вашого контенту, щоб створити зручну та логічну структуру таксономії.

Таксономія в Drupal 8 може бути використана для різних цілей та сценаріїв. Ось кілька варіантів використання таксономії в Drupal 8:

Категоризація контенту: Використання таксономії для категоризації контенту є одним з основних варіантів. Ви можете створити таксономічні терміни, які представляють різні категорії, теми або теги, і призначати їх до ваших вузлів або інших типів контенту. Це дозволяє користувачам фільтрувати та навігувати контент за цими категоріями.

Мітки та ключові слова: Таксономія може бути використана для створення міток або ключових слів, які допомагають описати та класифікувати контент. Наприклад, ви можете мати таксономію з ключовими словами, які відображають теми, теги або характеристики вашого контенту. Це полегшує пошук та орієнтацію на сайті.

Фільтрація та пошук: Завдяки таксономії, ви можете створювати фільтри та пошукові можливості, щоб дозволити користувачам швидко знаходити контент за певними категоріями, тегами або ключовими словами. Ви можете використовувати таксономію в пошукових формах, переглядах (Views) та інших елементах, що спрощує навігацію та пошук на сайті.

Меню та навігація: Drupal 8 надає можливість створювати меню на основі таксономії. Ви можете використовувати таксономічні терміни як пункти меню та підменю для структурування навігації на сайті. Це дозволяє створити ієрархічну навігацію засновану на вашій таксономії.

Розширення контенту: Таксономія може бути використана для розширення та класифікації контенту. Наприклад, ви можете мати таксономічне поле для вузлів, що дозволяє вам вибирати один або кілька таксономічних термінів, щоб прив'язувати контент до певних категорій або характеристик.

Покращена пошукова оптимізація: Використання таксономії дозволяє вам створювати ключові слова, теги та інші мітки, які покращують пошукову оптимізацію вашого сайту. Пошукові системи можуть краще індексувати та інтерпретувати контент, якщо він категоризований за допомогою таксономії.

Це лише кілька прикладів використання таксономії в Drupal 8. Залежно від вашого проекту та потреб, ви можете застосовувати таксономію для різних цілей та розширювати її функціонал за допомогою модулів та налаштувань Drupal.

Для налаштування форми редагування словника таксономії в Drupal 8 вам потрібно виконати наступні кроки:

Увійдіть до адміністративного інтерфейсу Drupal 8 і перейдіть до сторінки "Структура" > "Таксономія".

Виберіть словник таксономії, для якого ви хочете налаштувати форму редагування.

На сторінці управління таксономією виберіть опцію "Редагувати форму вводу" з випадаючого меню "Операції" поруч із відповідним словником.

На сторінці налаштування форми вводу для словника таксономії ви можете зробити наступне:

Додавання нових полів: На вкладці "Поля" ви можете додавати нові поля до форми редагування словника. Натисніть кнопку "Додати поле" і виберіть тип поля, який ви хочете додати. Налаштуйте параметри поля та збережіть зміни.

Порядок полів: Ви можете змінити порядок полів на вкладці "Поля", перетягуючи їх вгору чи вниз. Це визначає порядок, в якому поля відображаються на формі редагування.

Приховування полів: Якщо вам потрібно приховати певні поля на формі редагування словника, використайте вкладку "Поля" та встановіть параметр "Видимість" відповідного поля на значення "Приховано". Поля, які приховані, все ще будуть доступні для редагування через API, але не будуть відображатись на формі.

Обов'язковість полів: Ви можете налаштувати поле таксономії як обов'язкове, встановивши відповідне значення параметру "Обов'язкове" на вкладці "Поля". Це забезпечить, що користувачі мусять заповнити це поле перед збереженням форми.

Збереження налаштувань: Після внесення змін на сторінці налаштування форми вводу словника таксономії не забудьте натиснути кнопку "Зберегти", щоб зберегти зміни.

Таким чином, ви можете налаштувати форму редагування словника таксономії в Drupal 8, додавши, налаштувавши та керуючи полями, що використовуються на цій формі.

Для налаштування відображення термінів таксономії в Drupal 8 ви можете скористатись наступними кроками:

Увійдіть до адміністративного інтерфейсу Drupal 8 і перейдіть до сторінки "Структура" > "Таксономія".

Виберіть словник таксономії, для якого ви хочете налаштувати відображення термінів.

На сторінці управління таксономією виберіть опцію "Редагувати термін" з випадаючого меню "Операції" поруч із відповідним терміном.

На сторінці редагування терміна ви можете налаштувати його відображення. Основні налаштування включають наступні:

- Назва терміна: Змініть назву терміна, яка відображатиметься на сайті.
- URL-шлях: Можливість налаштувати URL-шлях терміна для SEO-оптимізації та зручного використання.
- Батьківський термін: Встановіть батьківський термін для створення ієрархії термінів, якщо необхідно.
- Опис терміна: Додайте опис, який пояснює термін або надає додаткову інформацію.
- Активність: Встановіть параметр "Активно" для включення або виключення терміна з відображення на сайті.
- Поле зображення: Якщо використовується модуль "Поле зображення", ви можете прикріпити зображення до терміна.
- Тип відображення: На сторінці редагування терміна ви також можете вибрати тип відображення, яке використовуватиметься для відображення терміна на сайті. Це залежить від вашої теми та налаштувань Drupal.

Після внесення змін на сторінці редагування терміна не забудьте натиснути кнопку "Зберегти", щоб зберегти зміни.

Таким чином, ви можете налаштувати відображення термінів таксономії в Drupal 8, змінюючи їх назву, URL-шлях, батьківські терміни, опис, активність та використовуючи налаштування типу відображення.

В Drupal 8 є вбудована підтримка перекладу термінів таксономії, що дозволяє створювати багатомовні таксономічні терміни. Особливості перекладу термінів таксономії в Drupal 8 включають наступне:

Модуль "Content Translation": Для перекладу термінів таксономії використовується модуль "Content Translation". Вам потрібно включити цей модуль, якщо він ще не включений, для використання функціоналу перекладу.

Конфігурація мов: Спочатку вам потрібно налаштувати мови вашого сайту. Перейдіть до адміністративного інтерфейсу Drupal 8, виберіть "Адміністрування" >

"Конфігурація" > "Мови" > "Мови сайту" і додайте потрібні мови, які ви хочете використовувати для перекладу термінів таксономії.

Включення перекладу для таксономії: Перейдіть до сторінки управління таксономією (Структура > Таксономія), виберіть потрібний словник таксономії і включіть опцію "Перекладати" для цього словника. Це дозволить вам перекладати терміни, пов'язані з цим словником.

Переклад термінів: Після включення перекладу для таксономії ви зможете перекладати окремі терміни. Перейдіть до сторінки редагування терміна (Структура > Таксономія > [словник] > Редагувати термін) і виберіть необхідну мову для перекладу. Ви зможете ввести переклади назви та інших полів терміна для обраної мови.

Використання модулів "Entity Translation" або "Entity Reference Translation": Якщо ви плануєте використовувати переклад термінів разом з перекладом сутностей (наприклад, переклад полів), вам можуть знадобитись додаткові модулі, такі як "Entity Translation" або "Entity Reference Translation". Ці модулі дозволяють перекладати всі аспекти сутностей, включаючи їх поля та посилання на таксономію.

Використання інструментів перекладу: Drupal 8 надає інструменти для управління перекладом, такі як модуль "Content Translation", інтерфейс перекладу і перекладові мовні пакети. Ці інструменти допомагають вам легко керувати перекладами та забезпечувати однорідність перекладів на вашому сайті.

Завдяки цим особливостям ви зможете ефективно перекладати терміни таксономії в Drupal 8 і створювати багатомовні сайти з гнучкою категоризацією контенту.

2.3.4. ПРОФІЛІ КОРИСТУВАЧІВ ТА ЇХ НАЛАШТУВАННЯ

Архітектура профілів користувачів в Drupal 8 базується на сутностях (entities) та політиках доступу (access policies). Основні компоненти архітектури профілів користувачів включають наступні:

Сутність "Користувач": У Drupal 8 кожен користувач представлений сутністю "Користувач" (User entity). Ця сутність зберігає основну інформацію про користувача, таку як ім'я, електронну пошту, пароль та інше.

Розширення сутності "Користувач": Drupal 8 надає можливість розширювати сутність "Користувач", додаваючи до неї додаткові поля. Це дозволяє створювати власні

поля для профілів користувачів, які зберігають додаткову інформацію про користувачів, наприклад, дату народження, адресу, контактні дані тощо.

Політики доступу до профілів: Drupal 8 використовує політики доступу для керування правами доступу до профілів користувачів. За допомогою політик доступу можна визначити, хто може переглядати, редагувати або видаляти профілі користувачів. Налаштування політик доступу дозволяє контролювати, яка інформація профілю доступна для різних ролей користувачів.

Модулі для розширення профілів: Drupal 8 надає можливість розширювати функціонал профілів користувачів за допомогою модулів. Існують модулі, які додають додаткові поля до профілів, модулі для налаштування політик доступу, а також модулі, що додають нові можливості для управління профілями користувачів.

Управління профілями через адміністративний інтерфейс: Drupal 8 надає адміністративний інтерфейс для управління профілями користувачів. Через цей інтерфейс ви можете переглядати, редагувати, видаляти та налаштовувати профілі користувачів.

API для роботи з профілями: Drupal 8 надає розширене API для роботи з профілями користувачів. Це дозволяє розробникам створювати власні модулі і розширювати функціонал профілів, працюючи зі сутностями, полями та політиками доступу.

Загалом, архітектура профілів користувачів в Drupal 8 надає гнучкість та розширюваність для створення та керування профілями користувачів, а також контролю доступу до їх інформації.

Правильне проектування профілів користувачів в Drupal 8 вимагає уважного планування та визначення вимог вашого проекту. Ось декілька кроків та рекомендацій, які можуть допомогти вам в цьому процесі:

Визначення вимог: Розібратися з вимогами вашого проекту щодо профілів користувачів. Це можуть бути додаткові поля, які потрібно включити в профілі, політики доступу до профілів, інтеграція з іншими модулями і т.д. Сформулюйте список вимог, щоб мати чітке уявлення про те, які функціональність та дані повинні бути в профілях користувачів.

Розробка типу профілю: В Drupal 8 використовується розширення сутності "Користувач" для створення профілів користувачів. Спочатку вам потрібно визначити, які поля ви хочете включити в профілі. Розгляньте, яку інформацію ви хочете зберігати

про користувачів, і створіть відповідні поля для типу профілю. Наприклад, ім'я, прізвище, адреса, дата народження тощо.

Додаткові поля профілю: Якщо потрібно додаткові поля для профілів, ви можете використати модуль "Field UI", щоб створити власні поля для сутності "Користувач". Розгляньте, які додаткові дані ви хочете зберігати, наприклад, контактні дані, соціальні мережі, інтереси тощо.

Політики доступу: Врахуйте, які права доступу потрібно надати користувачам для перегляду та редагування профілів. Drupal 8 використовує політики доступу для керування правами доступу до профілів користувачів. Розгляньте ролі користувачів, які повинні мати доступ до різних частин профілю, і налаштуйте політики доступу відповідно.

Кастомізація вигляду профілю: Drupal 8 надає можливість налаштовувати вигляд профілів користувачів за допомогою модуля "Views". Ви можете створити свої власні вигляди, які відображатимуть потрібну інформацію з профілів, а також налаштувати їх вигляд за допомогою шаблонів.

Модулі розширення: Drupal 8 має широкий вибір модулів, які додають додаткові функції та можливості для профілів користувачів. Розгляньте, які додаткові функції ви хочете включити в профілі, наприклад, можливість завантаження фотографій, соціальні мережі, коментарі, оцінки тощо, і встановіть відповідні модулі для цього.

Правильне проектування профілів користувачів в Drupal 8 залежить від унікальних вимог вашого проекту. Важливо враховувати потреби користувачів, функціональність, безпеку і зручність використання при створенні та налаштуванні профілів.

Для налаштування форми редагування профілю користувача в Drupal 8 ви можете скористатись наступними кроками:

Створення полів профілю: Перед тим як налаштовувати форму редагування профілю, спочатку вам потрібно створити відповідні поля для профілю користувача. Використовуйте модуль "Field UI" для створення необхідних полів, таких як текстові поля, випадаючі списки, фотографії тощо. Налаштуйте поля згідно ваших потреб.

Налаштування форми редагування: Після створення полів перейдіть до налаштування форми редагування профілю. Це можна зробити за допомогою модуля "Form UI". Його можна знайти на сторінці "Структура" > "Форми" у вашому адміністративному інтерфейсі.

Вибір форми редагування: За допомогою модуля "Form UI" ви зможете переглянути всі доступні форми редагування для користувачів. Знайдіть форму редагування профілю користувача (зазвичай називається "User profile form") і виберіть її для налаштування.

Налаштування полів на формі: На сторінці налаштування форми редагування ви зможете вибрати та налаштувати поля, які повинні бути включені на формі. Виберіть поля, які ви хочете відображати на формі редагування профілю, та налаштуйте їх параметри, такі як мітки, порядок, обов'язковість, тип відображення тощо.

Політики доступу: Розгляньте політики доступу до форми редагування профілю. Ви можете встановити права доступу до форми для різних ролей користувачів, включити або виключити можливість редагування певних полів залежно від ролей, а також встановити правила доступу до самої форми.

Збереження налаштувань: Після внесення змін на сторінці налаштування форми редагування не забудьте натиснути кнопку "Зберегти" для збереження налаштувань.

Після цих кроків форма редагування профілю користувача буде налаштована згідно вашого проекту. Користувачі зможуть вводити та змінювати дані профілю через цю форму.

Для налаштування відображення профілю користувача в Drupal 8 ви можете скористатись наступними кроками:

Створення режиму відображення: Почніть зі створення режиму відображення профілю користувача. В режимі відображення ви вказуєте, які поля та як їх відображати на сторінці профілю користувача. Перейдіть до адміністративного інтерфейсу Drupal і перейдіть до "Структура" > "Типи контенту" > "Користувач". Виберіть режим "Default" або створіть новий режим, якщо потрібно.

Налаштування полів: Налаштуйте, які поля повинні бути включені на сторінці профілю користувача. Виберіть поле, яке хочете налаштувати, і виберіть тип відображення для нього. Drupal 8 надає різні типи відображення, такі як "Default" (за замовчуванням), "Full content", "Teaser" та інші. Виберіть той, який відповідає вашим потребам.

Налаштування форматування та вигляду: Налаштуйте форматування та вигляд полів на сторінці профілю користувача. Ви можете використовувати HTML-теги, CSS-стилі та шаблони, щоб налаштувати вигляд полів так, як вам потрібно. Drupal 8 надає можливість використовувати шаблони та CSS-класи для кожного поля.

Збереження налаштувань: Після внесення змін на сторінці налаштування режиму відображення не забудьте натиснути кнопку "Зберегти" для збереження налаштувань.

Перевірка відображення: Перейдіть на сторінку профілю користувача, щоб перевірити, як відображаються змінені налаштування. Переконайтеся, що поля відображаються та форматуються так, як ви очікували.

За допомогою цих кроків ви можете налаштувати відображення профілю користувача в Drupal 8 згідно ваших потреб та дизайну вашого проекту.

2.3.5. БЛОКИ ТА ЇХ НАЛАШТУВАННЯ

У Drupal 8 блоки використовуються для відображення вмісту або функціональності на сторінках вашого сайту. Блоки можуть містити будь-який тип вмісту, такий як текст, зображення, посилання, форми або вбудовані модулі. Ось кілька ключових аспектів використання блоків у Drupal 8:

Створення блоків: Ви можете створювати власні блоки або використовувати готові блоки, які надаються модулями Drupal або темами. Щоб створити власний блок, ви можете скористатися модулем "Block" (базовий модуль Drupal) або створити власний модуль з власними блоками.

Розташування блоків: Блоки можна розміщувати на різних регіонах сторінок вашого сайту. Drupal 8 надає певну кількість регіонів за замовчуванням, таких як "Sidebar", "Header", "Footer" тощо, але ви можете додатково визначити свої власні регіони.

Керування відображенням блоків: Drupal 8 надає гнучкі налаштування для керування відображенням блоків на різних сторінках вашого сайту. Ви можете встановити видимість блоків на основі шляху, типу вмісту, ролей користувачів та інших умов.

Розширення можливостей блоків: Drupal 8 також надає можливості розширення блоків за допомогою модулів. Ви можете використовувати модулі для додавання нових типів блоків, кастомізації відображення блоків, використання контекстуальних фільтрів, кешування блоків та багато іншого.

Використання блоків у темах: Теми Drupal 8 можуть використовувати блоки для створення макетів сторінок. Ви можете визначати, які блоки відображаються у певних регіонах і як вони відображаються з використанням шаблонів теми.

В цілому, блоки є потужним інструментом управління вмістом та функціональністю вашого сайту у Drupal 8. Вони дозволяють вам гнучко контролювати відображення вмісту та інтерактивних елементів на сторінках вашого сайту.

Використання блоків в Drupal 8 має свої переваги і недоліки, які варто враховувати при розробці веб-сайту. Ось кілька основних переваг і недоліків:

Переваги використання блоків в Drupal 8:

Гнучкість в управлінні вмістом: Блоки дають можливість гнучко управляти вмістом і функціональністю на сторінках сайту. Ви можете додавати, редагувати та видаляти блоки на різних регіонах сторінок з легкістю, а також контролювати їх відображення залежно від умов.

Макети сторінок: Блоки дозволяють використовувати макети сторінок, щоб встановлювати розташування та відображення блоків у певних регіонах сторінок. Це дозволяє створювати різні макети для різних типів сторінок на вашому сайті.

Варіанти відображення: Drupal 8 надає можливість визначити кілька варіантів відображення одного блоку, наприклад, для різних мов або типів користувачів. Це дозволяє настроювати блоки для різних аудиторій вашого сайту.

Контекстуальні фільтри: Drupal 8 надає контекстуальні фільтри, що дозволяють встановлювати умови відображення блоків, залежно від контексту, такого як URL-адреси, типи вмісту, ролі користувачів тощо. Це дає вам більше контролю над тим, коли і де блоки будуть відображатися.

Можливість розширення: Ви можете створювати свої власні блоки або використовувати готові блоки, що надаються модулями Drupal або темами. Це дозволяє вам розширювати функціональність вашого сайту та пристосовувати його до ваших потреб.

Недоліки використання блоків в Drupal 8:

Обмежена гнучкість в розташуванні: Блоки в Drupal 8 розташовуються в певних регіонах сторінок, що може обмежувати вашу гнучкість у розташуванні елементів на сторінці. Наприклад, вам може бути складно точно визначити розташування блока всередині вмісту.

Обробка шаблонів: Якщо ви хочете повністю контролювати відображення блоків, вам може знадобитися маніпуляція шаблонами теми. Це вимагає додаткових знань і розуміння Drupal-тем і може бути викликом для новачків.

Перевантаження блоками: Надмірне використання блоків на сторінках може привести до перевантаження, особливо при великій кількості блоків або складному

вмісті. Це може погіршити продуктивність вашого сайту та сповільнити завантаження сторінок.

Потреба в керуванні: Багато блоків на сайті може призвести до потреби управління їх відображенням і конфігурацією. Це може стати проблемою, особливо при масштабуванні сайту або залученні декількох адміністраторів.

Вибір використання блоків у Drupal 8 залежить від специфічних потреб вашого проекту та вашого рівня досвіду з Drupal.

Проектування блоків в Drupal 8 включає кілька кроків, які допоможуть вам створити ефективні та гнучкі блоки для вашого веб-сайту. Ось кілька рекомендацій:

Визначте потреби та цілі: Ретельно проаналізуйте потреби вашого проекту та визначте, які функції або вміст повинен бути відображений у блоках. Визначте цілі, які ви хочете досягти за допомогою блоків (наприклад, привернення уваги користувачів до певних елементів, пропаганда акцій або відображення спеціального вмісту).

Виберіть правильну стратегію розташування: Розгляньте, де саме ви хочете розмістити блоки на вашому сайті. Drupal 8 надає певну кількість регіонів за замовчуванням, але ви також можете створювати власні регіони. При виборі регіонів враховуйте структуру вашого сайту та потреби користувачів.

Розділіть логіку та відображення: Розробка блоків в Drupal 8 базується на принципах модульності та повторного використання. Розділіть логіку блока (бізнес-логіка, запити до бази даних тощо) від його відображення. Використовуйте хуки або сервіси для обробки логіки та шаблони для відображення.

Використовуйте конфігураційні налаштування: Drupal 8 надає можливість використовувати конфігураційні налаштування для блоків. Використовуйте їх для настроювання поведінки блоків, таких як вибір змісту, фільтри, сортування тощо. Це дозволить вам легко змінювати параметри блоків без необхідності зміни коду.

Перевірте видимість та доступність: Враховуйте, коли і де ви хочете відображати блоки. Використовуйте умови видимості (такі як URL-адреси, типи вмісту, ролі користувачів тощо) для керування відображенням блоків. Переконайтеся, що ваші блоки доступні і відповідають потребам різних аудиторій.

Тестування та оптимізація: Після розробки блоків варто провести тестування для перевірки їх роботи та відображення на різних пристроях та браузерах. Переконайтеся, що ваші блоки працюють належним чином і не впливають на продуктивність вашого сайту. Оптимізуйте блоки, якщо це необхідно, щоб забезпечити їх ефективну роботу.

Документація: Завжди добре документувати ваші блоки, включаючи їх призначення, налаштування, розташування та будь-які специфічні вимоги. Це допоможе вам і іншим розробникам зрозуміти, як працюють ваші блоки та як їх змінювати в майбутньому.

Загалом, правильне проектування блоків в Drupal 8 полягає у ретельному плануванні, дотриманні кращих практик Drupal та врахуванні потреб вашого проекту та користувачів.

У Drupal 8 є вбудована підтримка перекладу блоків, що дозволяє вам створювати вміст на різних мовах та керувати його відображенням для кожної мови. Ось кілька особливостей перекладу блоків в Drupal 8:

Мультимовність: Drupal 8 має вбудовану систему мультимовності, що дозволяє створювати контент на різних мовах. Ви можете налаштувати список підтримуваних мов на вашому сайті та створювати переклади для кожного блоку на кожну мову.

Контентний переклад: Контент блоків може бути перекладений на різні мови. Ви можете створювати альтернативні версії блоків для кожної мови, включаючи текст, зображення, посилання тощо. Це дозволяє вам надавати зміст у відповідній мові для вашої аудиторії.

Відображення залежно від мови: Ви можете налаштовувати, які блоки будуть відображатися для кожної мови. Використовуйте налаштування видимості блоків, щоб вказати, які блоки будуть видимими для кожної мови. Наприклад, ви можете налаштувати блоки таким чином, щоб вони відображалися лише для певних мов або не відображалися взагалі.

Мовні зони та регіони: Drupal 8 дозволяє вам налаштовувати мовні зони та регіони для блоків. Ви можете визначити різні регіони для блоків на кожній мові, що дозволяє створювати різні макети та розташування блоків для різних мовних версій вашого сайту.

Мовні пакети та імпорт/експорт перекладів: Drupal 8 надає можливість створювати мовні пакети, що дозволяють ефективно керувати перекладами блоків. Ви можете імпортувати та експортувати переклади блоків для зручного управління мовами.

Загалом, переклад блоків в Drupal 8 дозволяє створювати багатомовні сайти та забезпечує гнучкість у керуванні вмістом для різних мовних версій. Ви можете змінювати та налаштовувати вміст блоків для кожної мови окремо, що дозволяє створювати персоналізований досвід для користувачів з різних мовних груп.

2.3.6. МЕНЮ ТА ЇХ НАЛАШТУВАННЯ

Меню в Drupal 8 використовується для навігації по сторінках вашого веб-сайту. Drupal 8 надає розширені можливості для створення та керування меню. Ось кілька ключових аспектів, пов'язаних з меню в Drupal 8:

Види меню: Drupal 8 дозволяє створювати різні види меню для різних цілей. Наприклад, ви можете мати основне меню, меню навігації по категоріях, меню швидкого доступу та інші. Кожен вид меню може мати свої властивості та налаштування.

Створення меню: Щоб створити нове меню, перейдіть до адміністративного інтерфейсу Drupal і відкрийте розділ "Меню". Там ви зможете створити нове меню, вказавши його назву та інші властивості. Також можна створити меню автоматично, використовуючи зареєстровані типи вмісту.

Додавання пунктів меню: Після створення меню ви можете додавати пункти меню. Кожний пункт меню має свою назву, шлях (URL) та інші атрибути. Ви можете налаштувати вкладеність пунктів меню, що дозволяє створювати ієрархічну структуру навігації.

Розташування меню: Drupal 8 надає регіони, де ви можете розміщувати ваші меню на сторінках сайту. Ви можете визначити регіони для відображення основного меню, меню навігації по категоріях, меню в підвалі тощо. Використовуйте налаштування теми або блоків для розташування меню на сторінцях.

Відображення меню: Ви можете налаштувати спосіб відображення меню з використанням шаблонів теми або модулів. Drupal 8 надає гнучкі можливості для налаштування вигляду меню, включаючи стилі, класи CSS, розкриваючі підменю тощо.

Керування видимістю: Drupal 8 дозволяє вам керувати видимістю меню на різних сторінках сайту. Ви можете вибрати, на яких сторінках меню буде відображатися, а на яких - приховуватися. Використовуйте налаштування видимості або контекстуальні фільтри для контролю відображення меню.

Переклад меню: Drupal 8 надає підтримку перекладу меню для багатомовних сайтів. Ви можете створювати переклади для кожного пункту меню та керувати їх відображенням для різних мов.

Загалом, меню в Drupal 8 дозволяє гнучко керувати навігацією вашого сайту. Ви можете створювати різні види меню, налаштовувати їх вигляд та видимість, а також перекладати для багатомовних сайтів.

Правильне проектування меню в Drupal 8 включає декілька кроків, які допоможуть створити ефективну та зручну навігацію на вашому веб-сайті. Ось кілька рекомендацій:

Аналіз потреб користувачів: Почніть з аналізу потреб вашої цільової аудиторії. Розуміння їх очікувань та способу, яким вони шукають інформацію, допоможе вам створити зручну та логічну структуру меню. Визначте основні розділи і підрозділи, які найкраще відповідають потребам користувачів.

Планування структури меню: Врахуйте логіку вашого контенту та взаємозв'язки між сторінками. Створіть ієрархічну структуру меню з основними пунктами та їх підрозділами. Використовуйте логічні та зрозумілі назви для кожного пункту меню, щоб користувачі легко зорієнтувалися.

Використання регіонів та розташування: Плануйте, де саме ви хочете розмістити ваші меню на сторінках сайту. Drupal 8 надає регіони, де можна розташовувати меню, такі як основна навігація, меню в підвалі тощо. Виберіть регіони, які найкраще відповідають вашим потребам та способу взаємодії з користувачами.

Додавання пунктів меню: Після створення меню додайте пункти до відповідних розділів. Використовуйте підрозділи, якщо це необхідно, для створення більшої ієрархії в навігації. Враховуйте логіку користувачів та спрощення доступу до важливої інформації.

Враховуйте адаптивний дизайн: Забезпечте, щоб ваше меню працювало на різних пристроях та розмірах екранів. Використовуйте адаптивні техніки, такі як згортання меню на мобільних пристроях, щоб забезпечити зручну навігацію для користувачів на будь-якому пристрої.

Тестування та оптимізація: Перевірте роботу вашого меню на різних сторінках сайту. Впевніться, що воно працює належним чином і забезпечує зручну навігацію для користувачів. Оптимізуйте меню, якщо це необхідно, щоб полегшити доступ до важливих розділів та покращити користувацький досвід.

Управління меню: Періодично оновлюйте ваше меню, додаючи нові пункти або видаляючи зайві. Використовуйте аналітику та зворотний зв'язок від користувачів для покращення вашої навігації.

Загалом, правильне проектування меню в Drupal 8 полягає у зрозумінні потреб користувачів, плануванні структури, використанні регіонів та розташування, тестуванні та оптимізації. Пам'ятайте про зручність користування та логіку навігації при створенні свого меню.

Використання меню в Drupal 8 має свої переваги та недоліки, які варто враховувати при розробці веб-сайту. Ось кілька ключових переваг і недоліків:

Переваги використання меню в Drupal 8:

Гнучкість: Drupal 8 надає велику гнучкість при створенні та керуванні меню. Ви можете створювати різні види меню з різними рівнями вкладеності та ієрархією.

Керованість: Меню в Drupal 8 легко керувати через адміністративний інтерфейс. Ви можете легко додавати, редагувати та видаляти пункти меню, а також налаштовувати їх видимість та розташування.

Багатомовність: Drupal 8 має вбудовану підтримку багатомовності, що дозволяє створювати переклади для пунктів меню. Ви можете налаштовувати відображення меню для різних мов та аудиторій.

Адаптивний дизайн: Drupal 8 дозволяє створювати адаптивне меню, що працює на різних пристроях і екранах. Ви можете налаштовувати вигляд меню для мобільних, планшетних та настільних пристроїв.

Недоліки використання меню в Drupal 8:

Обмежена гнучкість розташування: Drupal 8 має обмежені регіони для розташування меню. Це може обмежувати вашу гнучкість у розміщенні та стилізації меню на сторінці.

Складність налаштування: Якщо вам потрібно складніші налаштування або вигляд меню, вам може знадобитися додаткове вивчення і розуміння Drupal API та темізації.

Продуктивність: Велика кількість меню або складні ієрархії можуть вплинути на продуктивність вашого веб-сайту, особливо при використанні кешування.

Залежність від теми: Вигляд та стиль меню можуть бути обмеженими тим, яку тему ви використовуєте. Вам може знадобитися додаткова робота зі стилізацією, щоб досягти бажаного вигляду.

Загалом, використання меню в Drupal 8 дозволяє ефективно керувати навігацією на вашому веб-сайті. Проте, варто враховувати певні обмеження та складнощі, щоб належним чином налаштувати та використовувати меню.

Переклад меню в Drupal 8 дозволяє створювати багатомовні сайти та керувати відображенням меню для різних мов. Особливості перекладу меню в Drupal 8 включають наступне:

Мовні варіації: У Drupal 8 ви можете створювати мовні варіації для кожного пункту меню. Це означає, що ви можете мати різні тексти, URL-адреси та

налаштування для кожної мови. Кожна мовна варіація пов'язана зі своєю мовною версією меню.

Мовні зони: Drupal 8 дозволяє вам налаштовувати різні мовні зони для меню. Мовна зона визначає, які мови будуть використовуватися для конкретного меню. Ви можете визначити одну мовну зону для всіх мов або створити окрему мовну зону для кожної мови.

Переклади меню: Drupal 8 надає можливість створювати переклади для меню. Ви можете створювати переклади пунктів меню, які будуть відображатися для різних мов. Кожен пункт меню може мати свої власні переклади, які можна налаштовувати та редагувати.

Видимість на різних мовах: Ви можете налаштовувати видимість пунктів меню для кожної мови. Наприклад, ви можете налаштувати, щоб деякі пункти меню були видимими лише для певних мов, а інші - для всіх мов. Ви також можете встановлювати умови видимості для кожного пункту меню, щоб керувати його відображенням на різних сторінках.

Переклади URL-адресів: У разі потреби Drupal 8 дозволяє перекладати URL-адреси пунктів меню. Ви можете мати різні URL-адреси для кожної мови, що спрощує створення багатомовного сайту

Питання для самоконтролю:

1. Що таке сутність в термінології Drupal 8?
2. Що таке тип контенту в Drupal 8?
3. Що таке поля, форма, відображення? Як їх використовувати?
4. Що таке нода?
5. Що таке таксономія?
6. Які переваги та недоліки таксономії?
7. Що таке блоки?
8. Як перекладаються блоки?
9. Що таке сутність користувача, та як її використовувати?
10. Як використовується меню в Drupal 8?

2.4. КОНСТРУКТОР ЗАПИТІВ VIEWS

Views в Drupal 8 - це модуль, який надає потужні і гнучкі можливості для створення та керування списками вмісту на вашому веб-сайті. Використовуючи модуль Views, ви можете легко налаштовувати запити до бази даних і створювати кастомні списки вмісту, які можуть бути відображені на сторінках вашого сайту.

Основні особливості Views в Drupal 8:

Конфігураційний інтерфейс: Views має інтуїтивно зрозумілий конфігураційний інтерфейс, який дозволяє вам визначати параметри відображення списків вмісту. Ви можете встановлювати фільтри, сортування, поля, групування, обмеження, агрегацію та інші параметри для відображення вмісту.

Гнучкість та розширюваність: Views надає велику гнучкість у відображенні та обробці вмісту. Ви можете налаштовувати вигляд списків вмісту за допомогою різних режимів відображення, таких як таблиці, списки, сітки тощо. Крім того, ви можете створювати власні поля, фільтри, сортування та інші елементи для персоналізації відображення.

Підтримка контентних типів: Views інтегрується з контентними типами Drupal, дозволяючи вам створювати списки вмісту на основі різних типів контенту. Ви можете налаштовувати відображення списків вмісту для кожного типу контенту окремо, використовуючи різні поля та параметри.

Кешування та продуктивність: Views підтримує кешування, що допомагає покращити продуктивність вашого веб-сайту. Ви можете налаштовувати час кешування та залежності від інших елементів, щоб забезпечити швидку відповідь на запити.

Інтеграція з іншими модулями: Views інтегрується з іншими модулями Drupal, що дозволяє вам використовувати його функціональність у поєднанні з іншими модулями. Наприклад, ви можете поєднувати Views з модулем Panels для створення складних макетів сторінок.

Загалом, Views є потужним інструментом для створення та керування списками вмісту в Drupal 8. Ви можете легко налаштовувати відображення вмісту, створювати кастомні списки, розширювати функціональність та забезпечувати швидку продуктивність.

2.4.1. ЗАГАЛЬНІ ВІДОМОСТІ ПРО КОНСТРУКТОР ЗАПИТІВ VIEWS

Views в Drupal 8 працює на основі кількох основних компонентів, які співпрацюють між собою, щоб забезпечити створення та керування списками вмісту. Ось детальний опис процесу роботи Views:

Запит до бази даних: Перший крок полягає в складанні запиту до бази даних для отримання потрібного вмісту. Views створює SQL-запит, використовуючи параметри, вказані в інтерфейсі конфігурації. Запит може містити фільтри, сортування, обмеження, групування та інші умови.

Обробка результатів: Після виконання SQL-запиту, Views отримує результати з бази даних. Далі відбувається обробка результатів для форматування та відображення відповідного вмісту. Результати можуть бути оброблені функціями, які відповідають за форматування, фільтрацію, агрегацію та інші маніпуляції з даними.

Режим відображення: Views має різні режими відображення, які визначають, як саме вміст буде відображатися на сторінці. Режими відображення включають таблиці, списки, галереї, сітки та інші. Ви можете вибрати потрібний режим відображення в залежності від типу вмісту та зовнішнього вигляду, який ви бажаєте досягти.

Форматування та стилізація: Views дозволяє вам налаштовувати вигляд вмісту за допомогою різних налаштувань стилів та шаблонів. Ви можете застосовувати CSS-стилі, визначати класи та ідентифікатори для елементів, використовувати шаблони для кастомізації вигляду списків вмісту.

Кешування: Для поліпшення продуктивності, Views підтримує кешування результатів. Ви можете налаштовувати параметри кешування, такі як тривалість зберігання кешу, залежності від інших елементів тощо. Кешування дозволяє уникнути повторного формування списків вмісту при кожному завантаженні сторінки.

Інтеграція з іншими модулями: Views може інтегруватися з іншими модулями Drupal, що дозволяє розширити його функціональність. Наприклад, ви можете використовувати модуль Panels для створення складних макетів сторінок з використанням Views як контентного елемента.

Загалом, Views в Drupal 8 дозволяє вам легко створювати та керувати списками вмісту на вашому веб-сайті. Ви можете налаштовувати запити до бази даних, вибирати режим відображення, стилізувати та форматувати вміст, а також використовувати кешування для поліпшення продуктивності.

2.4.2. ПРИНЦИПИ ВИКОРИСТАННЯ VIEWS

Правильне використання Views в Drupal 8 включає кілька кроків, які допоможуть вам ефективно створювати та керувати списками вмісту на вашому веб-сайті.

Встановлення та активація модуля: Перш ніж почати використовувати Views, переконайтеся, що модуль Views встановлений та активований на вашому сайті. Це можна зробити через адміністративний інтерфейс Drupal в розділі "Manage modules" (Керування модулями).

Створення нового Views: Після активації модуля ви можете перейти до сторінки "Structure" (Структура) і вибрати "Views" для створення нового списку вмісту. Клацніть на кнопці "Add view" (Додати вид) для початку створення нового Views.

Налаштування параметрів Views: На сторінці налаштування Views ви можете визначити параметри відображення списку вмісту, такі як назву, опис, тип вмісту, фільтри, сортування, обмеження та інші налаштування. Виберіть потрібні параметри відповідно до ваших потреб та очікувань.

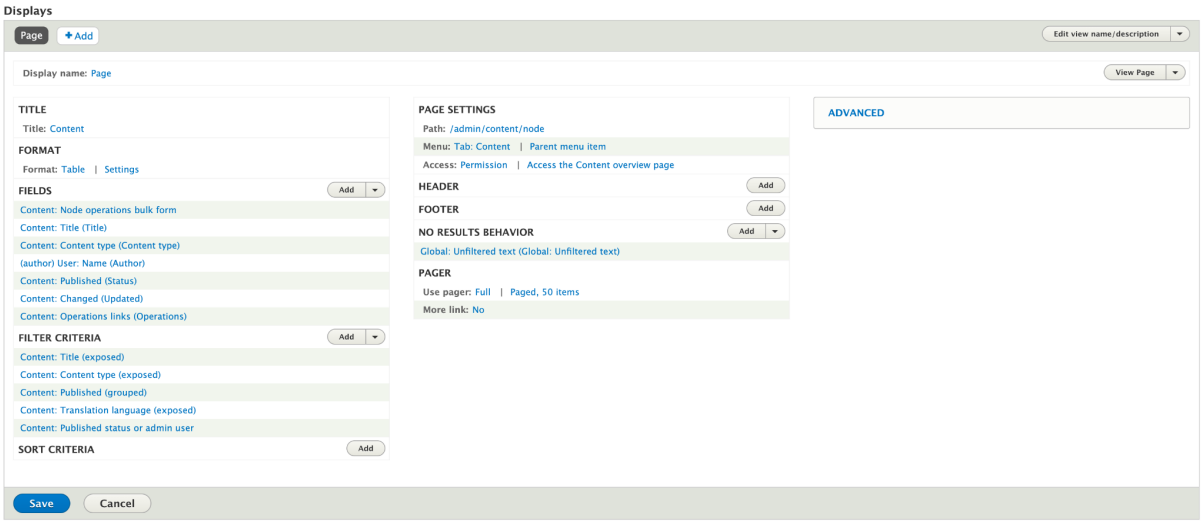

Рис. 4. Інтерфейс Views

Вибір режиму відображення: Після налаштування параметрів Views ви можете вибрати режим відображення для списку вмісту. Views надає різні режими, такі як таблиця, сітка, список, галерея тощо. Виберіть режим, який найкраще відповідає вашим потребам та візуальному стилю вашого сайту.

Форматування та відображення вмісту: Після вибору режиму відображення ви можете налаштувати форматування та вигляд списку вмісту. Використовуйте

налаштування стилів, шаблонів та CSS для відображення списку вмісту згідно з вашими потребами та дизайном.

Збереження та перегляд списку вмісту: Після завершення налаштування Views збережіть його та перегляньте результат. Ви можете переглянути список вмісту, вносити зміни, вимикати та включати Views за необхідності.

Інтеграція з іншими елементами сайту: Views може бути інтегрований з іншими елементами Drupal, такими як блоки, панелі, або використовуватися в макетах сторінок. Ви можете налаштовувати показ Views на різних сторінках сайту та використовувати його для створення складних макетів.

Тестування та оптимізація: Перевірте роботу вашого Views на різних сторінках сайту та переконайтеся, що воно працює належним чином. Оптимізуйте Views, якщо це необхідно, щоб полегшити доступ до важливих розділів та покращити користувацький досвід.

Загалом, правильне використання Views в Drupal 8 вимагає розуміння його основних параметрів та можливостей, а також уважного налаштування відповідно до потреб вашого веб-сайту. Пам'ятайте про гнучкість та можливості, які надає Views для створення та керування списками вмісту.

Також, варто пам’ятати, що кешування важливе у Views в Drupal 8 з кількох причин:

Покращення продуктивності: Кешування Views дозволяє зберегти результати запитів до бази даних та обробки вмісту, що дозволяє значно зменшити час завантаження сторінок. Кешовані результати можуть бути безпосередньо відправлені клієнту без потреби повторного формування списку вмісту.

Зменшення навантаження на сервер: Кешування дозволяє зменшити навантаження на сервер, оскільки він не потребує повторного формування списку вмісту для кожного запиту. Замість цього, кешовані результати можуть бути використані для швидкого відправлення клієнту.

Зниження кількості запитів до бази даних: Кешування дозволяє зменшити кількість запитів до бази даних, оскільки результати можуть бути збережені та використані безпосередньо. Це особливо корисно в ситуаціях, коли база даних має великий обсяг даних або знаходиться на віддаленому сервері.

Підтримка високої навантаженості: Кешування дозволяє забезпечити більш стабільну та швидку роботу сайту при високих навантаженнях. Коли багато

користувачів одночасно відвідує сторінки з використанням Views, кешовані результати можуть бути швидко відправлені, що допомагає зберегти продуктивність сайту.

Налаштування часу життя кешу: Drupal 8 дозволяє налаштовувати час життя кешу для Views. Ви можете встановити тривалість зберігання кешу, після якого дані будуть оновлюватися. Це дозволяє забезпечити актуальність вмісту та уникнути відображення застарілих даних.

Враховуючи ці фактори, важливо належним чином налаштувати та використовувати кешування у Views в Drupal 8. Розуміння та ефективне використання кешування допоможе покращити продуктивність вашого сайту та забезпечити швидке відображення списків вмісту.

2.4.3. ПОБУДОВА СТОРІНОК ТА БЛОКІВ З ВИКОРИСТАННЯМ VIEWS

При проектуванні сторінки на основі Views в Drupal 8, вам потрібно врахувати наступні кроки:

Визначте цільову аудиторію та мету сторінки: Розуміння, для кого ви створюєте сторінку та яку мету ви хочете досягти, є важливим вихідним пунктом. Визначте, які дані вам потрібні для відображення на сторінці і як вони пов'язані між собою.

Створіть новий View: Створіть новий View, який буде використовуватися для відображення списків вмісту на сторінці. Визначте налаштування, такі як тип вмісту, фільтри, сортування та інші параметри, які потрібні для відображення відповідного вмісту.

Налаштуйте вигляд та режим відображення: Виберіть відповідний режим відображення для вашого списку вмісту, такий як таблиця, сітка, список тощо. Налаштуйте вигляд та стилізацію вмісту за допомогою налаштувань CSS, шаблонів або модифікаторів класів.

Додайте поля та зв'язки: Виберіть необхідні поля для відображення на сторінці. Додайте зв'язки, якщо необхідно, для відображення додаткової інформації з інших сутностей або зв'язаних об'єктів.

Налаштуйте фільтри та сортування: Додайте фільтри, які дозволяють користувачам змінювати вміст на сторінці в залежності від їх потреб. Налаштуйте сортування, щоб дозволити користувачам змінювати порядок відображення списку вмісту.

Створіть блоки або віджети: Ви можете використовувати Views, щоб створювати блоки або віджети, які можна розмістити на різних областях сторінки. Налаштуйте параметри відображення блоків або віджетів, щоб вони відповідали вашим потребам.

Застосуйте Views до сторінки: Додайте створений Views до відповідної сторінки або макету за допомогою інструментів управління контентом або панелей. Розташуйте Views на сторінці відповідно до дизайну та логіки вашого проекту.

Тестуйте та оптимізуйте: Перевірте роботу сторінки, впевніться, що вміст відображається коректно та відповідає вашим очікуванням. Виконайте тестування на різних пристроях та перевірте швидкість завантаження сторінки. Оптимізуйте Views за необхідності, щоб поліпшити продуктивність та користувацький досвід.

Загалом, проектування сторінки на основі Views в Drupal 8 вимагає розуміння ваших потреб, використання відповідних налаштувань Views та інтеграцію з іншими елементами вашого сайту.

Для створення блоку у Views в Drupal 8, потрібно виконати наступні кроки:

Створіть новий View або відредагуйте наявний: У розділі "Structure" (Структура) вашого адміністративного інтерфейсу виберіть "Views". Створіть новий View або відредагуйте наявний View, який ви хочете перетворити на блок.

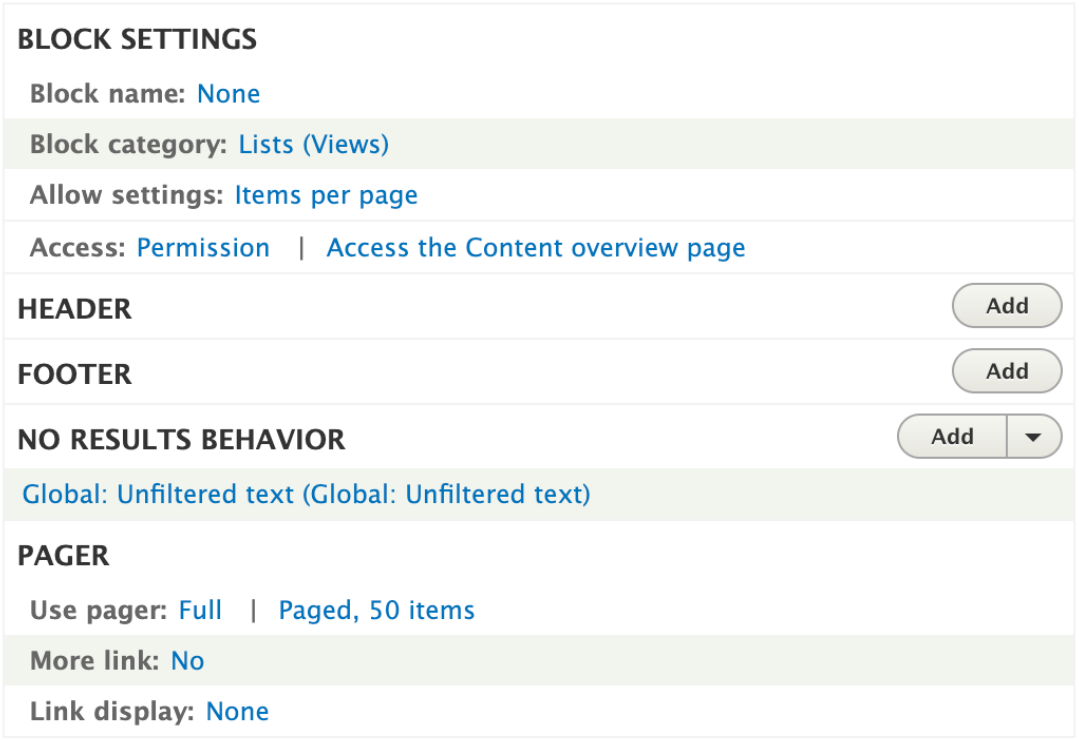

Рис. 5. Налаштуваннння параметрів блоку

Налаштуйте налаштування блоку: У вкладці "Advanced" (Розширені) сторінки конфігурації Views знайдіть розділ "Other" (Інше). Під параметром "Display format" (Формат відображення) виберіть "Block" (Блок).

Налаштуйте параметри блоку: Відкрийте налаштування блоку, натиснувши на посилання "Block settings" (Налаштування блоку) поряд із випадаючим списком "Block" у вкладці "Advanced". Тут ви можете встановити назву блоку, опис, розташування та інші параметри.

Збережіть налаштування та перевірте результат: Натисніть кнопку "Save" (Зберегти), щоб зберегти зміни в конфігурації Views. Потім перейдіть на веб-сайт і перевірте результат. Ви повинні бачити новий блок, який відповідає вказаним налаштуванням.

Розташуйте блок на сторінці: Щоб розмістити блок на сторінці, перейдіть до розділу "Structure" (Структура) вашого адміністративного інтерфейсу і виберіть "Block layout" (Розташування блоків). Знайдіть потрібне регіон на сторінці, куди ви хочете розмістити блок, і перетягніть блок зі списку доступних блоків до потрібного регіону.

Налаштуйте відображення блоку: Якщо ви бажаєте налаштувати відображення блоку, таке як стилізацію чи шаблон, ви можете використовувати тему вашого Drupal сайту або створити кастомні шаблони для блоку.

За допомогою цих кроків ви зможете створити та розмістити блок у Views в Drupal 8. Блок може бути відображений на сторінках вашого сайту згідно з обраною конфігурацією та розташуванням.

Кешування блоків у Views в Drupal 8 дозволяє зберігати результати відображення блоку для покращення продуктивності та швидкості завантаження сторінок. Кешування дозволяє уникнути повторного формування блоку при кожному завантаженні сторінки, особливо у випадку, коли вміст блоку змінюється рідко або не змінюється взагалі.

Ось деякі особливості кешування блоків у Views:

Встановлення параметрів кешування: Ви можете встановити різні параметри кешування для блоків у Views. Ці параметри включають тривалість зберігання кешу, залежності від інших об'єктів (наприклад, вмісту або конфігурації) та способи очищення кешу при зміні даних.

Використання контексту: У Views в Drupal 8 можна використовувати контекст для кешування блоків. Контекст дозволяє визначити, коли кеш блоку повинен бути використаний, а коли його потрібно перегенерувати. Наприклад, ви можете

використовувати контекст, щоб кешувати блок тільки для анонімних користувачів або тільки для конкретного типу вмісту.

Інструменти очищення кешу: Views надає інструменти для очищення кешу блоків у випадку, якщо вміст блоку або його контекст змінилися. Ви можете налаштувати автоматичне очищення кешу при певних подіях або використовувати інструменти для очищення кешу вручну.

Кастомне кешування: Ви можете налаштувати власне кастомне кешування для блоків у Views. Це дає вам більшу гнучкість та контроль над процесом кешування, дозволяючи вам вирішувати, коли та як кеш блоку повинен оновлюватися.

Вплив на продуктивність: Кешування блоків у Views може суттєво покращити продуктивність вашого сайту, особливо якщо ви маєте велику кількість вмісту або запитів до бази даних. Кешовані блоки можуть бути швидко відправлені клієнтам без необхідності повторного формування.

Щоб правильно використовувати кешування блоків у Views, ви повинні розуміти свої потреби, налаштувати параметри кешування відповідно до цих потреб та регулярно перевіряти та оптимізовувати свої налаштування кешування.

Views Attachments є потужним інструментом в Drupal 8, який дозволяє включати додатковий вміст або блоки на сторінку, пов'язану з головним списком вмісту, створеним за допомогою Views. Використання Views Attachments дозволяє створювати складніші сторінки, які містять декілька списків вмісту або додатковий контент, пов'язаний з основним списком.

Розглянемо основні аспекти використання Views Attachments в Drupal 8:

Створення основного списку вмісту: Спочатку ви повинні створити основний список вмісту за допомогою Views. Встановіть фільтри, сортування, поля та інші параметри, які відповідають вашим потребам. Це буде основа для відображення вашого контенту.

Створення нового додатку (Attachments): Додатки - це додаткові блоки або списки вмісту, які будуть включені на основну сторінку списку. Ви можете створити новий додаток, натиснувши на посилання "Add" (Додати) на сторінці налаштувань Views.

Налаштування параметрів додатка: На сторінці налаштувань додатка ви можете визначити параметри, такі як тип вмісту, фільтри, сортування, поля та інші налаштування, що стосуються додатку. Ці параметри можуть відрізнятися від основного списку вмісту, залежно від вашої потреби.

Встановлення зв'язку між основним списком та додатком: Щоб пов'язати додаток з основним списком, використовуйте функцію "Attachment settings" (Налаштування додатка) на сторінці налаштувань додатка. Ви можете вибрати, до якого місця на основній сторінці списку вмісту буде включено додаток (наприклад, до заголовку, футера або певної зони).

Встановлення вигляду та режиму відображення додатка: На сторінці налаштувань додатка ви також можете вибрати вигляд та режим відображення додатка, такі як таблиця, сітка або інші варіанти. Ви можете налаштувати стилізацію та вигляд додатка залежно від вашого дизайну та потреб.

Додавання додатка до основної сторінки списку: Після налаштування додатка ви можете додати його до основної сторінки списку вмісту. Залежно від вашої потреби, ви можете розташувати додаток у відповідному місці, наприклад, перед або після основного списку вмісту, в заголовку або футері сторінки.

Тестування та оптимізація: Перевірте результати, перегляньте сторінку з основним списком та додатком, щоб переконатися, що вони відображаються належним чином. Виконайте тестування на різних пристроях та перевірте швидкість завантаження сторінки. При необхідності оптимізуйте налаштування Views та додатку, щоб покращити продуктивність та користувацький досвід.

За допомогою Views Attachments в Drupal 8, ви можете створювати багатошарові сторінки з додатковим вмістом та списками вмісту, що допомагає покращити організацію ваших даних та надає більш гнучкий дизайн для вашого сайту.

Використання Views Attachments в Drupal 8 має свої переваги та недоліки, які варто враховувати при розробці сайту. Ось деякі з них:

Переваги використання Views Attachments:

Гнучкість та розширюваність: Views Attachments дозволяє створювати складніші сторінки, які містять додатковий контент або списки вмісту, пов'язані з основним списком. Ви можете включати додаткові блоки, поля або фільтри, щоб покращити функціональність та візуальний вигляд сторінки.

Контроль над відображенням: Ви маєте повний контроль над тим, як додаток відображається на сторінці. Ви можете вибрати вигляд, режим відображення та налаштувати стилізацію, щоб відповідати вашому дизайну та вимогам.

Перевикористання вмісту: Views Attachments дозволяє використовувати один додаток на кількох сторінках, що дозволяє ефективно використовувати ваш вміст та зменшити дублювання.

Недоліки використання Views Attachments:

Складність налаштування: Використання Views Attachments може бути складним для новачків, оскільки вимагає розуміння налаштувань Views та зв'язків між списками вмісту та додатками. Враховуйте, що правильне налаштування може зайняти деякий час та вимагати технічних знань.

Перфоманс: Використання багатьох додатків на одній сторінці може призвести до збільшення завантаження сторінки та витрат пам'яті сервера. Необхідно обережно використовувати Views Attachments та оптимізувати сторінки для забезпечення швидкої роботи та продуктивності.

Складність супроводу: При використанні Views Attachments велика кількість додатків та залежностей може зробити код більш складним та важким для супроводу. Враховуйте це при розробці та документуванні коду.

Незважаючи на деякі недоліки, Views Attachments є потужним інструментом в Drupal 8, який дозволяє створювати багатошарові сторінки та забезпечує більшу гнучкість та контроль над відображенням вашого контенту.

2.4.4. ВИКОРИСТАННЯ РЕЖИМІВ ВІДОБРАЖЕННЯ У VIEWS

Views Format - це одна з ключових можливостей модуля Views в Drupal 8, яка визначає спосіб відображення вмісту на сторінці. Використання формату дозволяє змінювати вигляд та структуру вмісту, а також налаштовувати його відображення залежно від потреб проекту. Детальніше про Views Format:

Формати відображення: Views Format надає різні формати відображення, такі як список, таблиця, полівка, сітка, розумний або карусель. Кожен формат визначає, як саме буде відображатися вміст, які поля включаються, як вони впорядковуються та як стилізуються.

Налаштування вигляду: Кожен формат відображення має свої налаштування вигляду, які дозволяють вам змінювати стилі, класи, межі, фон, шрифти та багато іншого. Ви можете використовувати CSS-класи, шаблони або інші методи для стилізації відображення вмісту залежно від вашого дизайну та потреб.

Доступність та SEO: Views Format дозволяє оптимізувати вміст для доступності та пошукових систем. Ви можете додавати атрибути, метатеги, заголовки, посилання та інші SEO-оптимізації до відображення вмісту, щоб покращити його індексацію та відтворення пошуковими системами.

Конфігурація полів: Views Format дозволяє вибирати, які поля включати до відображення та як їх форматувати. Ви можете налаштувати формат дати, часу, числа, вирівнювання тексту, маскування, стилізацію та інші параметри для кожного поля.

Контекстні фільтри та умови: Views Format дозволяє встановлювати контекстні фільтри та умови, що впливають на відображення вмісту. Ви можете використовувати фільтри для відображення лише певного вмісту, який відповідає заданим умовам, або для встановлення залежностей між вмістом у списку.

Додаткові налаштування: Ви можете налаштовувати багато інших параметрів у Views Format, таких як пагінація, сортування, ліміти, підписи, переклади, агрегати, виправлення помилок та багато іншого. Це дозволяє вам налаштувати відображення вмісту залежно від вашої потреби.

Використання Views Format у Drupal 8 дозволяє вам повністю контролювати вигляд та структуру вашого вмісту на сторінках, створених за допомогою Views. Ви можете створювати різні вигляди, налаштовувати стилізацію, оптимізувати для доступності та SEO та налаштовувати відображення залежно від потреб вашого проекту.

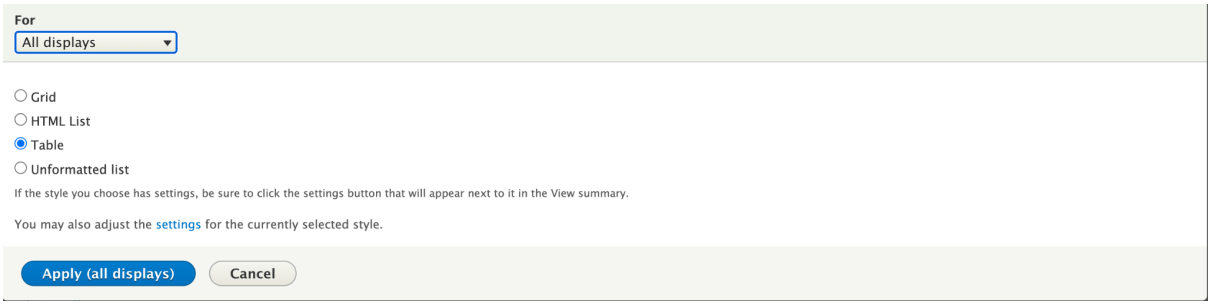

Рис. 6. Налаштування формату виводу

Правильний вибір формату виводу у Views в Drupal 8 залежить від типу вмісту, його структури та потреб вашого проекту. Ось кілька критеріїв, які можуть допомогти вам обрати правильний формат виводу:

Тип вмісту: Розгляньте тип вмісту, який ви виводите. Якщо це списковий вміст, такий як статті, новини або товари, формати виводу, які підтримують список або сітку, можуть бути більш підходящими. Якщо це окремі сторінки або детальні відомості, то формати виводу, які дозволяють більш докладне відображення, можуть бути корисними.

Вигляд та структура: Враховуйте бажаний вигляд та структуру вмісту. Якщо вам потрібен простий список або таблиця, то формати виводу, такі як "Unformatted list" або

"Table", можуть бути відповідними. Якщо ви хочете більш гнучкого та кастомізованого відображення, формати виводу, які дозволяють використовувати шаблони та розміщувати поля по-різному, можуть бути корисними.

Кількість полів та вмісту: Розгляньте кількість полів та вмісту, яке ви хочете відобразити. Якщо у вас є багато полів або велика кількість вмісту, формати виводу, які підтримують пагінацію або безкінечну прокрутку, можуть допомогти покращити навігацію та продуктивність.

Дизайн та стиль: Вибирайте формати виводу, які відповідають вашому дизайну та стилю. Деякі формати виводу мають вбудовані стилі та класи, які легко налаштовувати за допомогою CSS. Інші формати виводу можуть бути більш гнучкими, дозволяючи вам повністю контролювати стилізацію через власні шаблони або CSS-класи.

SEO та доступність: При виборі формату виводу враховуйте його вплив на SEO та доступність. Деякі формати виводу можуть забезпечувати оптимізацію для пошукових систем, надавати атрибути тегів та правильну структуру. Доступність також важлива, тому обирайте формати, які дозволяють користувачам з особливими потребами легко отримувати доступ до вмісту.

Нарешті, експериментуйте з різними форматами виводу та налаштуваннями, щоб знайти той, який найкраще підходить для вашого проекту. Запитуйте зворотний зв'язок від користувачів та прослуховуйте їхні потреби для подальшого вдосконалення відображення вмісту.

Формат виводу у вигляді полів (Fields) в Views в Drupal 8 має свої переваги та недоліки, які варто враховувати при використанні. Ось деякі з них:

Переваги формату виводу у вигляді полів:

Гнучкість та контроль: Формат виводу у вигляді полів дозволяє вам максимально контролювати вигляд та розміщення окремих полів вмісту на сторінці. Ви можете вибирати, які поля включати, в якому порядку вони відображаються та як їх стилізувати. Це дозволяє вам створювати настраювані та унікальні вигляди для свого вмісту.

Полегшена обробка даних: Формат виводу у вигляді полів дозволяє вам працювати з окремими полями вмісту, що полегшує обробку та маніпулювання даними. Ви можете застосовувати форматування, обрізку тексту, фільтрацію та інші операції до окремих полів вмісту залежно від вашої потреби.

Швидкодія: Формат виводу у вигляді полів може бути більш ефективним за рахунок мінімального обсягу даних, які потрібно витягнути з бази даних. Ви можете вибрати лише необхідні поля, уникнувши зайвих запитів та навантаження на сервер.

Недоліки формату виводу у вигляді полів:

Більше налаштувань: Використання формату виводу у вигляді полів може потребувати більше налаштувань та кастомізації, оскільки вам доведеться окремо налаштовувати кожне поле вмісту. Це може зайняти більше часу та зусиль для налаштування ідеального вигляду.

Складність для новачків: Формат виводу у вигляді полів може бути складним для новачків, оскільки вимагає розуміння структури та взаємодії полів вмісту. Необхідно мати базові знання про полі типів вмісту та їх налаштування.

Велика кількість полів: Якщо у вас є велика кількість полів вмісту, використання формату виводу у вигляді полів може збільшити складність та розмір конфігураційних файлів, що може вплинути на продуктивність та керованість проекту.

Оцінюйте свої потреби, розумійте переваги та недоліки формату виводу у вигляді полів та використовуйте його, коли він найкраще відповідає вимогам вашого проекту та спрощує вашу роботу з вмістом.

Розглянемо ще два поширені формати.

У Views в Drupal 8 формати виводу "Grid" (сітка) та "Table" (таблиця) мають різницю в способі відображення вмісту на сторінці.

Формат виводу "Grid" (сітка):

- Сітка представляється у вигляді блоків або карточок, розміщених у горизонтальних та вертикальних рядках.
- Кожен елемент в сітці може містити різні поля вмісту, такі як зображення, заголовок, опис, посилання тощо.
- Має гнучкі налаштування для кількості колонок та розмірів блоків.
- Зазвичай використовується для створення сітчастих сторінок зі списком вмісту, наприклад, галереї, товарів або портфоліо.

Формат виводу "Table" (таблиця):

- Таблиця представляється у вигляді табличної структури з рядками та колонками.
- Кожен рядок таблиці відповідає окремому запису вмісту, а кожна колонка представляє поля вмісту.

- Можна включати поля вмісту, такі як назва, автор, дата, категорія тощо.
- Має можливість сортування даних по колонках та пагінацію.
- Зазвичай використовується для табличних представлень даних, наприклад, списків користувачів, коментарів або продуктів.

Різниця між форматами полягає в способі відображення та організації вмісту. "Grid" забезпечує гнучкість у розміщенні та вигляді елементів, що дозволяє створювати затребувані сітчасті макети. "Table" надає структуроване табличне представлення даних з можливістю сортування та пагінації. Вибір формату залежить від потреб вашого проекту та способу відображення вмісту на сторінці

У Views в Drupal 8 формати виводу "Unformatted list" (неформатований список) та "HTML list" (HTML-список) мають різницю в способі відображення та розмітці вмісту на сторінці.

Формат виводу "Unformatted list" (неформатований список):

- Вміст відображається як простий список без додаткової розмітки або стилів.
- Кожен елемент списку може містити в собі дані з окремих полів вмісту, такі як заголовок, дата, опис тощо.
- Не має додаткових контейнерів або стилів, що дозволяє більшу гнучкість у стилізації та налаштуванні вигляду за допомогою CSS.

Формат виводу "HTML list" (HTML-список):

- Вміст відображається у вигляді HTML-списку з використанням відповідних HTML-тегів, таких як `<ul>` (ненумерований список) або `<ol>` (нумерований список).
- Кожен елемент списку може містити дані з окремих полів вмісту, які можуть бути розміщені у внутрішніх тегах, таких як `<li>`.
- Має вбудовану структуру списку, яка може бути корисною для стилізації та налаштування вигляду списку за допомогою CSS.

Різниця між форматами полягає в способі відображення та розмітці вмісту. "Unformatted list" надає простий неформатований список без додаткової розмітки, що дає більшу гнучкість у стилізації. "HTML list" надає структурований HTML-список з вбудованою розміткою, що полегшує стилізацію та налаштування вигляду списку за допомогою CSS. Вибір формату залежить від потреб вашого проекту та способу відображення вмісту на сторінці.

.

2.4.5. СТАТИЧНІ ТА ДИНАМІЧНІ ФІЛЬТРИ У VIEWS

У Views в Drupal 8 фільтри використовуються для обмеження вмісту, який відображається на сторінці, відповідно до заданих умов. Фільтри дозволяють вам вибирати конкретні записи або встановлювати умови, за якими відбувається відбір вмісту. Ось детальніше про фільтри у Views:

Вбудовані фільтри: Views надає ряд вбудованих фільтрів, таких як фільтр по полю, фільтр по автору, фільтр по даті тощо. Ви можете вибрати поле, по якому потрібно зробити фільтрацію, та задати умови, щоб відфільтрувати вміст на основі цих умов.

Контекстні фільтри: Крім вбудованих фільтрів, Views дозволяє використовувати контекстні фільтри. Контекстні фільтри залежать від контексту або значень з інших полів, які використовуються у вашому перегляді. Вони дозволяють вам встановлювати залежності між полями та створювати більш складні фільтраційні умови.

Комбіновані фільтри: Ви можете комбінувати декілька фільтрів, використовуючи логічні оператори AND та OR, для створення більш складних фільтраційних правил. Наприклад, ви можете використовувати фільтр по полю "Категорія" AND фільтр по полю "Дата" для відображення записів, які належать до певної категорії та були опубліковані після певної дати.

Власні фільтри: Views дозволяє створювати власні фільтри, які відповідають вашим специфічним потребам. Ви можете використовувати власні PHP-коди для створення фільтра, який базується на складних умовах або користувацьких даних.

Фільтри в Views дозволяють вам налаштовувати відображення вмісту, враховуючи потреби вашого проекту та критерії фільтрації. Ви можете встановлювати умови, які відповідають конкретним значенням полів, датам, категоріям тощо, або використовувати більш складні фільтраційні правила для отримання бажаного вмісту.

Налаштування фільтрів у Views в Drupal 8 залежить від конкретних потреб вашого проекту та критеріїв фільтрації. Ось деякі кроки, які можуть допомогти вам правильно налаштувати фільтри:

Виберіть поле для фільтрації: Виберіть поле вмісту, яке ви хочете використовувати для фільтрації. Наприклад, це може бути поле "Категорія", "Дата" або будь-яке інше поле, яке ви хочете використовувати як критерій фільтрації.

Встановіть умови фільтрації: Визначте умови, за якими ви хочете відфільтрувати вміст. Наприклад, ви можете встановити умову, що поле "Категорія" повинно мати певне значення або що поле "Дата" повинно бути в межах певного діапазону.

Використовуйте логічні оператори: Якщо ви хочете використовувати більш складні фільтраційні правила, використовуйте логічні оператори AND та OR. Наприклад, ви можете встановити умову, що поле "Категорія" повинно мати певне значення AND поле "Дата" повинно бути після певної дати.

Використовуйте контекстні фільтри (за бажанням): Контекстні фільтри дозволяють вам фільтрувати вміст залежно від значень інших полів у вашому перегляді. Вони корисні, коли потрібно встановити залежності між полями для точнішої фільтрації.

Перевірте попередній перегляд: Після налаштування фільтрів перегляньте попередній перегляд, щоб переконатися, що вміст відображається так, як ви очікуєте. Переконайтеся, що фільтри працюють належним чином і відображають лише відповідний вміст.

Налаштуйте додаткові параметри (якщо потрібно): Views також надає додаткові параметри для фільтрів, такі як сортування, пагінація, відображення повного тексту або скороченого зображення. Налаштуйте ці параметри залежно від вашого проекту та вимог до вмісту.

Пам'ятайте, що налаштування фільтрів у Views в Drupal 8 є гнучким процесом, і ви можете експериментувати з різними умовами та налаштуваннями, щоб отримати бажаний результат. Перевіряйте та тестуйте ваші фільтри, щоб переконатися, що вони працюють належним чином і задовольняють ваші потреби.

У Views в Drupal 8 ви можете комбінувати фільтри, використовуючи логічні оператори, такі як AND та OR, для створення більш складних фільтраційних правил. Ось кілька кроків, які допоможуть вам комбінувати фільтри:

Додайте перший фільтр: Створіть перший фільтр за допомогою вбудованого функціоналу Views або створеного вами власного фільтра. Встановіть умови фільтрації для цього фільтра залежно від вашого критерію.

Додайте другий фільтр: Додайте другий фільтр, використовуючи той самий процес, як і для першого фільтра. Встановіть умови фільтрації для цього фільтра відповідно до вашого другого критерію.

Виберіть логічний оператор: Виберіть логічний оператор, який ви хочете використовувати для комбінації фільтрів. Наприклад, якщо ви хочете відображати вміст, який задовольняє обидва критерії, використовуйте оператор AND. Якщо ви хочете відображати вміст, який задовольняє хоча б один з критеріїв, використовуйте оператор OR.

Налаштуйте комбінований фільтр: Використовуйте логічний оператор для комбінування фільтрів у вашому перегляді. Встановіть логічний оператор для відповідного місця в конфігурації фільтру.

Перевірте попередній перегляд: Після налаштування комбінованого фільтру перегляньте попередній перегляд, щоб переконатися, що вміст відображається за відповідними фільтраційними умовами.

Налаштуйте додаткові параметри (за бажанням): Налаштуйте інші параметри для фільтрів, такі як сортування, пагінація або додаткові налаштування фільтрів, залежно від вашого проекту та вимог до вмісту.

Пам'ятайте, що комбінація фільтрів у Views дає вам можливість створювати більш складні фільтраційні правила для відображення вмісту. Експериментуйте з різними умовами та комбінаціями фільтрів, щоб отримати бажаний результат. Перевіряйте і тестуйте ваші фільтри, щоб переконатися, що вони працюють належним чином і задовольняють ваші потреби.

Exposed filters в Views в Drupal 8 дозволяють користувачам динамічно фільтрувати вміст на сторінці перегляду. Вони створюють форму з фільтрами, яку користувач може заповнити для вибору конкретних значень фільтрації. Ось кроки, які допоможуть вам використовувати exposed filters у Views:

Додайте фільтр до вашого перегляду: Створіть або виберіть фільтр, який ви хочете використовувати для exposed filters. Це може бути поле вмісту або будь-яке інше поле, за яким ви хочете здійснювати фільтрацію.

Встановіть параметри exposed filters: У налаштуваннях фільтру встановіть параметр "Expose this filter to visitors, to allow them to change it" на "Yes". Це дозволить відобразити фільтр у формі на сторінці перегляду.

Налаштуйте параметри exposed filters: Виберіть параметри для вашого exposed filter, такі як його назву, місце розташування на сторінці та вигляд.

Перегляньте результати: Перевірте попередній перегляд вашого перегляду, щоб переконатися, що exposed filter з'явився на сторінці і працює належним чином. Ви можете заповнити форму з фільтрами та переглянути результати, які відповідають введеним значенням.

Налаштуйте додаткові параметри (за бажанням): Views також надає додаткові параметри для exposed filters, такі як сортування, пагінація, відображення повного тексту або скороченого зображення. Налаштуйте ці параметри залежно від вашого проекту та вимог до вмісту.

Exposed filters дозволяють вам дати користувачам можливість самостійно фільтрувати вміст у вашому перегляді. Вони створюють інтерактивну та персоналізовану зона для фільтрації, що покращує взаємодію з вашим сайтом та надає більш гнучкий спосіб взаємодії з вмістом.

Правильна комбінація exposed filters в Views в Drupal 8 дозволяє створити потужні інтерактивні фільтри для вашого вмісту. Ось декілька кроків, які допоможуть вам правильно комбінувати exposed filters:

Додайте фільтри до вашого перегляду: Створіть або виберіть фільтри, які ви хочете використовувати для exposed filters. Вони можуть бути полями вмісту, такими як "Категорія", "Тег", "Дата" тощо.

Налаштуйте параметри exposed filters: У налаштуваннях кожного фільтру встановіть параметр "Expose this filter to visitors, to allow them to change it" на "Yes". Це дозволить відобразити фільтр у формі на сторінці перегляду.

Виберіть тип зв'язку між фільтрами: При комбінуванні декількох exposed filters виберіть тип зв'язку між ними. Це може бути логічний оператор AND або OR. Наприклад, використовуйте оператор AND, якщо вам потрібно, щоб обидва фільтри були задоволені, або оператор OR, якщо вам потрібно, щоб хоча б один з фільтрів був задоволений.

Налаштуйте позиціонування exposed filters: Розмістіть exposed filters на вашій сторінці перегляду відповідно до вашого дизайну та потреб. Ви можете розмістити їх у верхній частині, бічній панелі або навіть вбудувати їх у вміст сторінки.

Перевірте попередній перегляд: Після налаштування exposed filters перегляньте попередній перегляд, щоб переконатися, що фільтри працюють належним чином. Заповнюйте форму з фільтрами та переглядайте вміст, що відповідає введеним значенням.

Налаштуйте додаткові параметри (за бажанням): Налаштуйте додаткові параметри для exposed filters, такі як сортування, пагінація або додаткові налаштування фільтрів, залежно від вашого проекту та вимог до вмісту.

Комбінування exposed filters дозволяє користувачам точно визначати, який вміст вони хочуть бачити. Завдяки цьому ваші користувачі матимуть змогу взаємодіяти з вмістом, відповідним їхнім власним критеріям та вимогам.

Існує доповнення до типового функціоналу - Better Exposed Filters (BEF) - це модуль для Drupal 8, який доповнює функціонал exposed filters у Views та надає додаткові можливості для кастомізації та поліпшення користувацького інтерфейсу

фільтрів. Ось кілька особливостей та переваг використання модуля Better Exposed Filters:

Стилізація та вигляд: BEF дозволяє вам легко налаштовувати стилізацію та вигляд фільтрів. Ви можете задати класи CSS, шаблони для кастомного відображення, вибрати типи полів (радіокнопки, прапорці, випадаючий список) та налаштувати інші параметри для кращого візуального вигляду.

Збереження значень фільтрів: BEF дозволяє зберігати значення фільтрів між сторінками. Це дозволяє користувачам зберігати свої налаштування фільтрів навіть після переходу на іншу сторінку або після повернення на попередню сторінку.

Додаткові елементи керування: BEF додає додаткові елементи керування для фільтрів, такі як кнопки "Скинути" або "Застосувати", що полегшують користувачам управління фільтрами та їхнім застосуванням.

Візуальні покажчики стану фільтрів: BEF дозволяє відображати візуальні покажчики, які показують, які фільтри використовуються, а також їхні значення. Це полегшує користувачам розуміння того, які фільтри вже застосовані та які значення вони мають.

Додаткові можливості фільтрації: BEF надає додаткові можливості фільтрації, такі як можливість застосування багатьох значень до одного фільтру, фільтрація за допомогою URL-параметрів, автоматичне оновлення вмісту без перезавантаження сторінки та інші.

Загалом, Better Exposed Filters розширює можливості exposed filters у Views, дозволяючи вам краще керувати фільтрацією та поліпшити користувацький досвід у вашому Drupal 8 проекті.

Аргументи в Views в Drupal 8 дозволяють передавати параметри у вигляді значень до вашого перегляду для динамічного фільтрування вмісту. Ви можете використовувати аргументи для створення більш гнучких та динамічних переглядів. Ось кілька кроків, які допоможуть вам працювати з аргументами в Views:

Додайте аргумент до вашого перегляду: В розділі "Advanced" (Розширено) вашого перегляду перейдіть до розділу "Contextual Filters" (Контекстні фільтри) і додайте новий аргумент, натиснувши на кнопку "Add" (Додати). Виберіть поле або тип даних, яке ви хочете використовувати як аргумент.

Налаштуйте параметри аргументу: В налаштуваннях аргументу встановіть параметри, такі як джерело даних аргументу (наприклад, поле вмісту, путь URL або

інші), режим обробки аргументу (наприклад, за замовчуванням, з URL або інші) і додаткові параметри, які ви хочете використовувати.

Налаштуйте відображення значень аргументу: Ви можете встановити, як аргумент повинен трактувати передані значення. Наприклад, ви можете використовувати аргумент як фільтр для відображення вмісту, в якому поле співпадає з переданим значенням аргументу.

Передавайте значення аргументу: Щоб передати значення аргументу до вашого перегляду, ви можете використовувати різні способи, включаючи передачу значень через URL-параметри, контекстний фільтр або програматично в коді вашої сторінки.

Перевірте результати: Після налаштування аргументу перегляньте результати, щоб переконатися, що вміст відображається залежно від переданих значень аргументу.

Аргументи в Views дозволяють вам створювати більш динамічні та гнучкі перегляди, які можуть залежати від зовнішніх параметрів або даних. Ви можете використовувати їх для створення фільтрів, пошуку, сортування та багато іншого.

Використання значень за замовчуванням в аргументах Views в Drupal 8 дозволяє встановити певне значення аргументу, якщо воно не передається зовнішнім джерелом (наприклад, через URL або контекстний фільтр). Це дає вам можливість контролювати вміст, який відображається за замовчуванням, якщо не передано іншого значення. Ось кілька кроків, які допоможуть вам використовувати значення за замовчуванням в аргументах Views:

Додайте аргумент до вашого перегляду: В розділі "Advanced" (Розширено) вашого перегляду перейдіть до розділу "Contextual Filters" (Контекстні фільтри) і додайте новий аргумент, натиснувши на кнопку "Add" (Додати). Виберіть поле або тип даних, яке ви хочете використовувати як аргумент.

Встановіть значення за замовчуванням: У налаштуваннях аргументу є параметр "Default Value" (Значення за замовчуванням). Виберіть потрібний варіант значення за замовчуванням, наприклад, "Fixed Value" (Фіксоване значення), "Raw Value from URL" (Необроблене значення з URL) або інші варіанти в залежності від вашого випадку використання.

Налаштуйте параметри значення за замовчуванням: Встановіть значення за замовчуванням аргументу відповідно до вашого вимоги. Наприклад, якщо ви обрали "Fixed Value", введіть фіксоване значення, яке ви хочете використовувати як аргумент за замовчуванням.

Перевірте результати: Після налаштування значення за замовчуванням перегляньте результати, щоб переконатися, що вміст відображається залежно від встановленого значення за замовчуванням, коли не передається інше значення.

Значення за замовчуванням в аргументах Views дозволяють вам контролювати вміст, який відображається за замовчуванням, і забезпечують більш гнучке керування вашими переглядами.

2.4.6. НАЛАШТУВАННЯ АГРЕГАЦІЇ ДАНИХ У VIEWS

Використання значень за замовчуванням в аргументах Views в Drupal 8 дозволяє встановити певне значення аргументу, якщо воно не передається зовнішнім джерелом (наприклад, через URL або контекстний фільтр). Це дає вам можливість контролювати вміст, який відображається за замовчуванням, якщо не передано іншого значення. Ось кілька кроків, які допоможуть вам використовувати значення за замовчуванням в аргументах Views:

Додайте аргумент до вашого перегляду: В розділі "Advanced" (Розширено) вашого перегляду перейдіть до розділу "Contextual Filters" (Контекстні фільтри) і додайте новий аргумент, натиснувши на кнопку "Add" (Додати). Виберіть поле або тип даних, яке ви хочете використовувати як аргумент.

Встановіть значення за замовчуванням: У налаштуваннях аргументу є параметр "Default Value" (Значення за замовчуванням). Виберіть потрібний варіант значення за замовчуванням, наприклад, "Fixed Value" (Фіксоване значення), "Raw Value from URL" (Необроблене значення з URL) або інші варіанти в залежності від вашого випадку використання.

Налаштуйте параметри значення за замовчуванням: Встановіть значення за замовчуванням аргументу відповідно до вашого вимоги. Наприклад, якщо ви обрали "Fixed Value", введіть фіксоване значення, яке ви хочете використовувати як аргумент за замовчуванням.

Перевірте результати: Після налаштування значення за замовчуванням перегляньте результати, щоб переконатися, що вміст відображається залежно від встановленого значення за замовчуванням, коли не передається інше значення.

Значення за замовчуванням в аргументах Views дозволяють вам контролювати вміст, який відображається за замовчуванням, і забезпечують більш гнучке керування вашими переглядами.

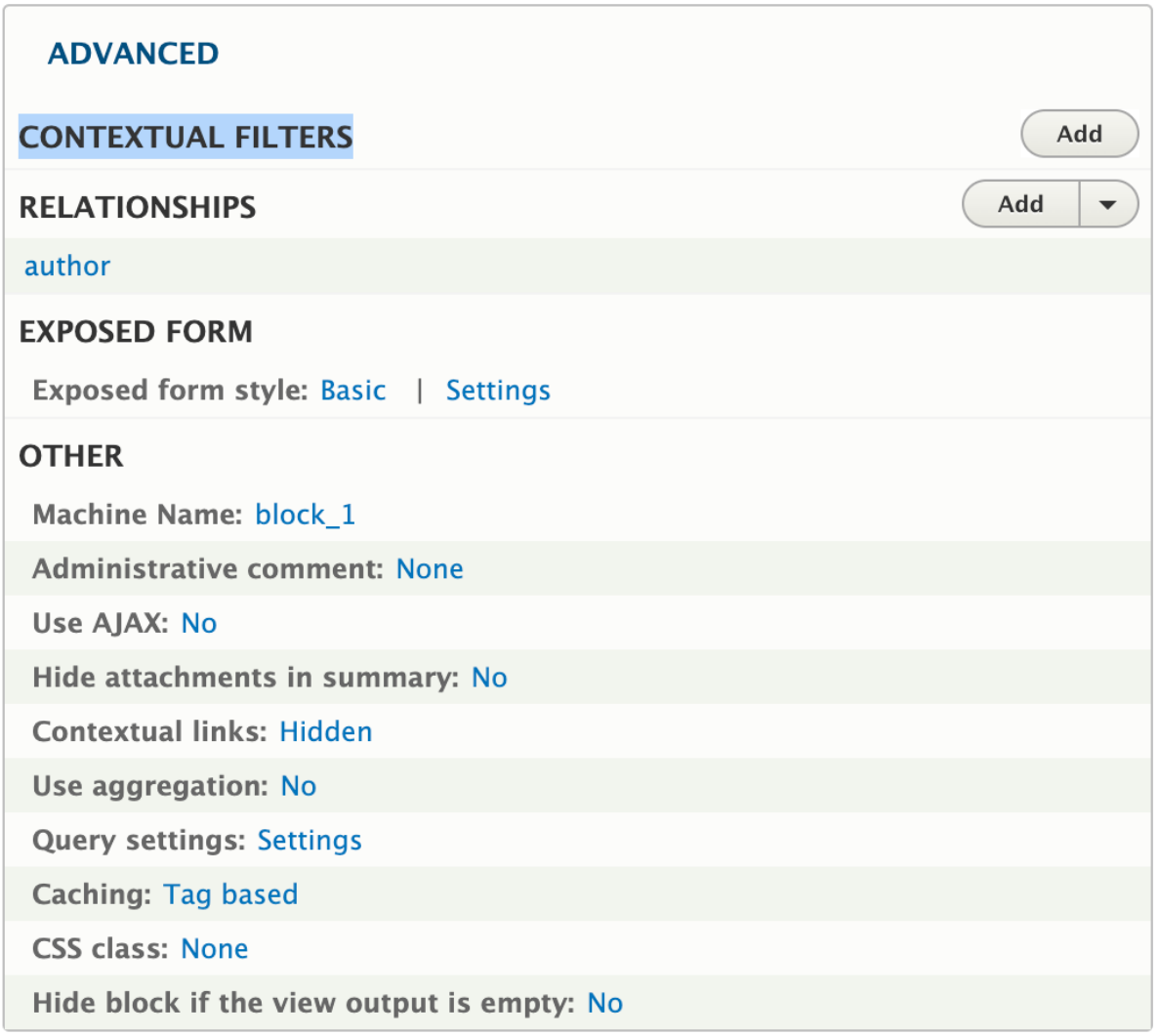

Рис. 7. Керування додатковими налаштуваннями Views

Налаштування "Disable SQL rewriting" та "Distinct" у Views в Drupal 8 використовуються для керування тим, як генерується SQL-запит для отримання даних з бази даних.

Disable SQL rewriting: За замовчуванням, Drupal використовує SQL-переписування (SQL rewriting), щоб додати до SQL-запиту додаткові умови, такі як безпека ролей користувачів, контекст аутентифікації, контекст доступу до вмісту та інші. Однак, у деяких випадках може бути потрібно вимкнути цей механізм для власних маніпуляцій з SQL-запитом. Якщо ви вимикаєте налаштування "Disable SQL

rewriting", ви забезпечуєте повний контроль над SQL-запитами, які використовуються в Views.

Distinct: Налаштування "Distinct" визначає, чи потрібно використовувати операцію DISTINCT в SQL-запиті. DISTINCT видаляє дублікати з результатів запиту, тобто кожен рядок буде унікальним. Якщо ви встановлюєте налаштування "Distinct" в значення "Yes", то SQL-запит буде містити операцію DISTINCT, що дозволить отримати лише унікальні результати. Це може бути корисно, наприклад, коли ви отримуєте повторювані рядки через зв'язки між таблицями або внаслідок складних умов фільтрації.

Обидва налаштування, "Disable SQL rewriting" та "Distinct", використовуються для кастомізації SQL-запитів у Views і дозволяють вам більш точно контролювати отримання даних з бази даних.

Relationships (зв'язки) у Views в Drupal 8 використовуються для встановлення зв'язку між різними таблицями або сутностями в базі даних. Вони дозволяють вам отримувати дані з пов'язаних таблиць і використовувати їх для відображення, фільтрації, сортування та групування вмісту в переглядах. Ось декілька прикладів того, як і для чого використовуються relationships у Views:

Відображення поля з пов'язаної сутності: Ви можете використовувати relationships, щоб відобразити поля з пов'язаних таблиць або сутностей. Наприклад, якщо ви маєте перегляд вмісту "Коментарі" і хочете відобразити поле "Автор коментаря" з пов'язаної сутності "Користувач", ви можете додати зв'язок до сутності "Користувач" і відобразити поле автора.

Фільтрація за допомогою пов'язаних полів: З використанням relationships ви можете фільтрувати вміст за допомогою полів з пов'язаних таблиць або сутностей. Наприклад, якщо ви хочете відфільтрувати вміст "Відео" за категорією, яка зберігається у пов'язаній таблиці "Категорії відео", ви можете додати зв'язок до таблиці "Категорії відео" і встановити фільтр за полем "Категорія".

Групування або сортування за пов'язаними полями: Ви можете групувати або сортувати вміст за допомогою полів з пов'язаних таблиць або сутностей. Наприклад, якщо ви хочете групувати "Замовлення" за країною клієнта, яка зберігається у пов'язаній таблиці "Клієнти", ви можете додати зв'язок до таблиці "Клієнти" і встановити групування за полем "Країна".

Побудова складних звітів або статистики: Завдяки relationships ви можете побудувати складні звіти або статистику, які використовують дані з декількох

пов'язаних таблиць або сутностей. Ви можете об'єднати дані з різних джерел і розрахувати загальні суми, середні значення, кількість або інші агреговані значення для аналізу або відображення.

Relationships дозволяють вам розширити можливості переглядів, створювати більш складні та зручні перегляди даних і працювати з пов'язаними таблицями або сутностями.

Кешування у Views в Drupal 8 дозволяє зберігати результати запитів до бази даних та компонування вигляду, щоб зменшити навантаження на сервер і збільшити швидкість завантаження сторінок. Ось детальний опис, як використовувати і налаштовувати кешування у Views:

Увімкніть кешування для перегляду: В розділі "Advanced" (Розширено) перегляду виберіть "Caching" (Кешування) і увімкніть опцію "Use Caching" (Використовувати кешування). Це дозволить включити кешування для цього перегляду.

Виберіть тип кешування: Ви можете вибрати один з трьох типів кешування: "None" (Ні), "Time-based" (За часом) або "Tag-based" (За тегами). "None" вимикає кешування, "Time-based" кешує результати на певний проміжок часу, а "Tag-based" дозволяє кешувати результати залежно від певних тегів.

Налаштуйте параметри кешування: Залежно від вибраного типу кешування, ви можете налаштувати додаткові параметри. Наприклад, для "Time-based" кешування ви можете встановити час життя кешу, після якого він буде оновлюватися. Для "Tag-based" кешування ви можете додати теги, які будуть використовуватися для інвалідації кешу при зміні відповідних даних.

Налаштуйте працю кешування для динамічних елементів: Якщо ви маєте динамічні елементи, такі як фільтри або аргументи, які можуть впливати на вміст перегляду, ви можете налаштувати їх вплив на кешування. В розділі "Advanced" перегляду виберіть "Caching" (Кешування) і налаштуйте параметри "Cache Metadata" (Метадані кешу) для врахування динамічних елементів при кешуванні.

Збережіть налаштування і перевірте результати: Після налаштування кешування збережіть налаштування перегляду і перевірте результати. Зауважте, що після зміни даних або налаштувань, які впливають на вміст перегляду, кеш буде оновлюватися, щоб відображати актуальну інформацію.

Налаштування і використання кешування у Views дозволяють покращити швидкість завантаження сторінок і знизити навантаження на сервер. Проте, слід пам'ятати, що кешування повинно бути налаштоване з обережністю, особливо для

динамічного вмісту, оскільки воно може призводити до відображення застарілих даних у разі несвоєчасної інвалідації кешу.

Views може також використовуватись і для побудови складних форматів виводу. Для цього зазвичай потрібно використовувати або кастомні або контрібні модулі.

Views Data Export - це модуль Drupal 8, який дозволяє експортувати дані з переглядів у різних форматах файлів, таких як CSV, XLS, XML, JSON і т.д. Він надає можливість створювати спеціальні перегляди для експорту даних і налаштовувати формати виведення для цих даних.

Ось кілька ключових можливостей та процес роботи з Views Data Export:

Встановлення та активація модуля: Спочатку ви повинні встановити та активувати модуль Views Data Export. Це можна зробити за допомогою Drupal UI або за допомогою Composer.

Створення перегляду для експорту даних: Після активації модуля ви можете створити новий перегляд або редагувати існуючий для налаштування експорту даних. Виберіть тип перегляду, який відповідає вашим потребам (наприклад, перегляд вмісту, перегляд користувачів і т.д.).

Додавання експортного поля: У налаштуваннях перегляду додайте поле, яке ви хочете експортувати. Наприклад, якщо ви хочете експортувати поля вмісту, додайте відповідні поля з таблиці "Вміст".

Налаштування формату виведення: Перейдіть до налаштувань формату виведення у вашому перегляді. Виберіть формат виведення, такий як CSV, XLS, XML, JSON і т.д. Налаштуйте параметри формату виведення, такі як роздільник, заголовки стовпців, шаблони, налаштування стилю і т.д.

Збереження налаштувань та перегляд результатів: Збережіть налаштування вашого перегляду та виконайте його. Ви отримаєте файл експорту вибраного формату з даними, які відповідають вашим налаштуванням.

Views Data Export дозволяє легко експортувати дані з Drupal у різних форматах, що дозволяє подальше аналізування, оброблення або обміну цими даними з іншими системами. Ви можете налаштувати виведення та форматування даних згідно з вашими потребами, щоб отримати потрібну структуру та оформлення експортованих даних.

Views Bulk Operations (VBO) - це модуль Drupal, який розширює можливості модуля Views, дозволяючи виконувати масові операції над вмістом або сутностями, відображеними в перегляді. Він дозволяє вам виконувати дії, такі як видалення,

оновлення, виконання масових дій або будь-яких інших налаштованих дій над вибраними елементами.

Ось кілька ключових можливостей та процес роботи з Views Bulk Operations:

Встановлення та активація модулю: Спочатку ви повинні встановити та активувати модуль Views Bulk Operations. Це можна зробити за допомогою Drupal UI або за допомогою Composer.

Створення перегляду з підтримкою VBO: Створіть новий перегляд або редагуйте існуючий перегляд, до якого ви хочете додати масові операції. Додайте необхідні поля, фільтри та сортування для відображення бажаного вмісту.

Додавання VBO-полів: У налаштуваннях перегляду додайте VBO-поля, які представляють доступні дії для вибраних елементів. Наприклад, додайте поле "VBO operations" (Операції VBO), в якому ви можете вибрати дії, такі як "Видалити", "Оновити" і т.д.

Конфігурація операцій: Налаштуйте параметри для кожної масової операції, яку ви хочете виконати. Наприклад, для операції "Видалити" ви можете налаштувати підтвердження видалення, повідомлення про успішне видалення і т.д.

Збереження налаштувань та перегляд результатів: Збережіть налаштування вашого перегляду та виконайте його. Ви побачите вміст з включеними масовими операціями, які ви можете виконувати для вибраних елементів.

Views Bulk Operations дозволяє вам здійснювати швидкі та ефективні масові операції над вашим вмістом або сутностями, що дозволяє зменшити час і зусилля при редагуванні або управлінні великим обсягом даних. Він також може бути налаштований для виконання налаштованих дій, що розширює його можливості відповідно до вашого конкретного використання.

Питання для самоконтролю:

1. Що таке views?
2. Як налаштувати вивід у вигляді таблиці або списку у views?
3. Для чого використовується режим агрегації даних у views?
4. Як налаштувати фільтрацію даних у views?
5. Як налаштувати кешування у views?

2.5. КЕРУВАННЯ ПРАВАМИ ДОСТУПУ

У Drupal 8, концепція управління доступом базується на системі ролей та дозволів. Ця система надає гнучкий та розширюваний механізм для керування тим, які користувачі мають доступ до різних функцій та ресурсів вашого сайту. Основні компоненти концепції управління доступом в Drupal 8 включають наступні:

Ролі користувачів (Roles): Ролі визначають групи користувачів з певними привілеями та правами доступу. Drupal має попередньо встановлені ролі, такі як адміністратор, редактор, анонімний користувач та зареєстрований користувач. Ви можете створювати власні ролі користувачів з різними наборами дозволів.

Дозволи (Permissions): Дозволи визначають, які дії можуть бути виконані користувачами з певною роллю. Наприклад, дозволи можуть дозволяти доступ до сторінок адміністрування, можливість створювати, редагувати або видаляти контент, управління модулями, настройка тем та інші функції.

Розділена адміністрація (Separation of Administration): У Drupal 8 існує принцип розділення адміністрації, що означає, що адміністративні завдання і доступ до них можуть бути обмеженими окремими дозволами. Наприклад, ви можете дати користувачам роль "Редактор", яка дозволяє їм створювати і редагувати контент, але не надавати повний доступ до адміністративної панелі.

RBAC (Role-Based Access Control): Drupal 8 також підтримує RBAC, який дозволяє встановлювати дозволи на основі ролей і відносин між ними. Це забезпечує більш деталізоване та гнучке управління доступом для складних сценаріїв.

Використовуючи ці компоненти, ви можете налаштовувати рівні доступу для користувачів на вашому сайті, забезпечуючи контроль над тим, що вони можуть робити і яку інформацію вони можуть переглядати. Це дозволяє створювати різні ролі для різних типів користувачів і забезпечувати безпеку та конфіденційність даних.

Концепція управління доступом в Drupal 8 має кілька переваг і недоліків, які варто враховувати при розробці веб-сайту. Ось деякі з них:

Переваги:

Гнучкість: Drupal 8 надає гнучку систему ролей і дозволів, що дозволяє настроювати рівні доступу для різних типів користувачів. Ви можете встановлювати дозволи для доступу до конкретних сторінок, функцій, модулів та вмісту залежно від потреб вашого сайту.

Розділена адміністрація: Drupal 8 дозволяє розділити адміністративні завдання і надавати обмежений доступ до адміністративних функцій окремим користувачам чи

групам користувачів. Це дозволяє забезпечити безпеку і уникнути непередбачуваних помилок при редагуванні вмісту та налаштуванні сайту.

RBAC: Використання RBAC в Drupal 8 дозволяє налаштовувати складні сценарії доступу, встановлювати дозволи на основі ролей та відносин між ними. Це дозволяє більш детально контролювати доступ до функціональності і ресурсів вашого сайту.

Недоліки:

Складність налаштування: Управління доступом в Drupal 8 може бути складним для новачків або некваліфікованих користувачів. Вибір правильних дозволів, ролей та налаштувань може бути заплутаним завданням, особливо для веб-сайтів зі складною структурою.

Заплутаність ієрархії: В навігації дозволів і ролей може виникати заплутаність через складну ієрархію та взаємозв'язки між ними. Це може призводити до неточностей або проблем у встановленні правильних дозволів для користувачів.

Ризик безпеки: Некоректне налаштування дозволів може призвести до ризику безпеки. Недостатньо обмежений доступ може дозволити несанкціонованому користувачеві отримати доступ до конфіденційної інформації або змінити налаштування сайту. З іншого боку, надмірно обмежений доступ може заважати легітимним користувачам у виконанні їхніх завдань.

Однак, з правильним розумінням і налаштуванням, концепція управління доступом в Drupal 8 забезпечує потужні можливості для контролю над доступом до вашого веб-сайту.

2.5.1. СТРУКТУРА ОБЛІКОВИХ ЗАПИСІВ КОРИСТУВАЧІВ

Структура облікового запису користувача в Drupal 8 включає різні поля та властивості, що дозволяють зберігати та керувати інформацією про користувача. Основні елементи структури облікового запису користувача в Drupal 8 включають наступні:

Ім'я користувача (Username): Ім'я, за яким користувач буде ідентифікуватися на сайті. Ім'я користувача є унікальним для кожного облікового запису.

Пароль (Password): Хешований пароль, який дозволяє користувачеві автентифікуватися на сайті. Drupal 8 використовує хешування паролів для забезпечення безпеки.

Електронна пошта (Email): Адреса електронної пошти, пов'язана з обліковим записом користувача. Вона використовується для сповіщень, відновлення пароля та інших комунікаційних цілей.

Ролі (Roles): Користувач може бути призначений до однієї або декількох ролей, які визначають його привілеї та права доступу. Ролі дозволяють групувати користувачів за функціональними або організаційними характеристиками.

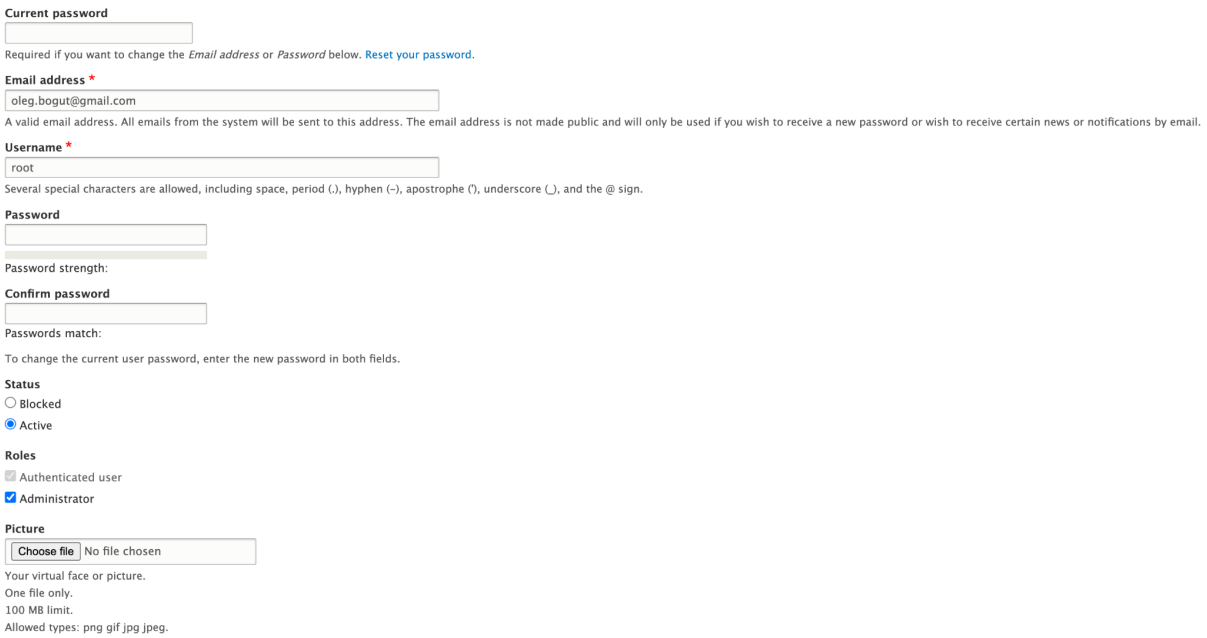

Рис. 8. Структура облікового запису користувача

Профіль (Profile): Drupal 8 має систему профілів, яка дозволяє розширити стандартну структуру облікового запису користувача за допомогою додаткових полів та властивостей. Ви можете створювати та налаштовувати власні поля профілю, що відповідають потребам вашого сайту.

Дозволи (Permissions): Користувачеві призначаються дозволи, які визначають, які дії він може виконувати на сайті. Дозволи контролюють доступ до функцій, ресурсів та модулів сайту.

Мета-інформація (Metadata): Обліковий запис користувача може містити додаткову мета-інформацію, таку як дата реєстрації, дата останнього входу, IP-адреса, статус активності тощо.

Структура облікового запису користувача в Drupal 8 може бути розширена та настроєна за допомогою модулів та налаштувань Drupal, що дозволяє вам

використовувати та зберігати додаткову інформацію про користувачів відповідно до вашого веб-сайту.

Для налаштування облікових записів користувачів в Drupal 8 ви можете використовувати наступні кроки:

Налаштування політики паролю:

- Перейдіть до "Configuration" (Конфігурація) у панелі адміністратора Drupal.
- У розділі "People" (Люди) виберіть "Account settings" (Налаштування облікових записів).
- В цьому розділі ви знайдете опції для налаштування політики паролю, такі як мінімальна довжина паролю, вимоги до складності, термін дії паролю тощо.

Створення ролей:

- Перейдіть до "People" (Люди) у панелі адміністратора Drupal.
- У розділі "Roles" (Ролі) виберіть "Add role" (Додати роль).
- Задайте назву та опис ролі і збережіть.

Налаштування дозволів:

- Перейдіть до "People" (Люди) у панелі адміністратора Drupal.
- У розділі "Permissions" (Дозволи) виберіть "Roles" (Ролі).
- Виберіть роль, для якої ви хочете налаштувати дозволи.
- Встановіть потрібні дозволи для цієї ролі та збережіть.

Налаштування полів профілю:

- Перейдіть до "Structure" (Структура) у панелі адміністратора Drupal.
- У розділі "Content types" (Типи контенту) виберіть "Manage fields" (Керування полями).
- Виберіть "User" (Користувач) для налаштування полів профілю користувача.
- Додайте або налаштуйте поля профілю за допомогою доступних типів полів.

Налаштування форми реєстрації:

- Перейдіть до "Configuration" (Конфігурація) у панелі адміністратора Drupal.

- У розділі "People" (Люди) виберіть "Account settings" (Налаштування облікових записів).
- Врозділі "Registration and cancellation" (Реєстрація та відмова) налаштуйте параметри форми реєстрації користувача, такі як обов'язкові поля, розміщення елементів тощо.

Це лише загальні кроки, і налаштування облікових записів користувачів в Drupal 8 може бути більш специфічним залежно від ваших потреб та вимог веб-сайту.

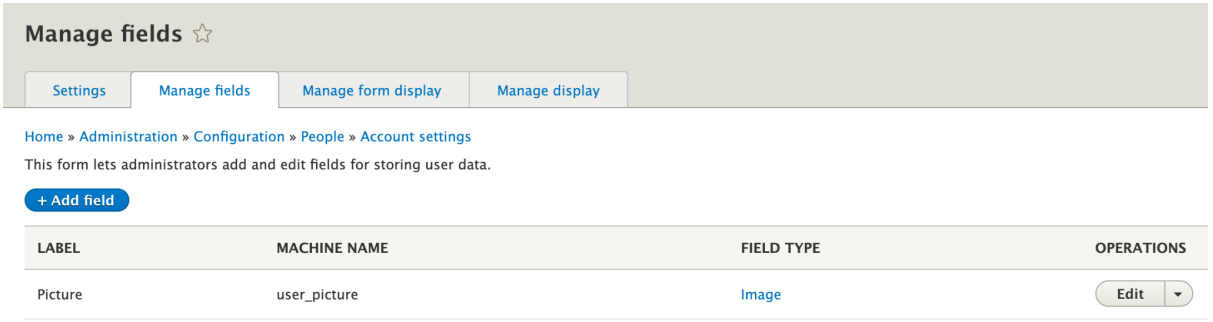

Рис. 9. Управління структурою облікового запису користувача

Для правильного керування полями в обліковому записі користувача в Drupal 8, ви можете виконати наступні кроки:

Налаштування полів профілю:

- Перейдіть до "Structure" (Структура) у панелі адміністратора Drupal.
- У розділі "Content types" (Типи контенту) виберіть "Manage fields" (Керування полями).
- Виберіть "User" (Користувач) для налаштування полів профілю користувача.
- Додайте нові поля або налаштуйте існуючі поля за допомогою доступних типів полів.

Встановлення обов'язковості полів:

- У налаштуваннях поля ви можете встановити, чи є поле обов'язковим для заповнення користувачем.
- Обов'язкові поля вимагатимуть від користувача вводу даних перед збереженням облікового запису.

Налаштування видимості полів:

- Ви можете налаштувати, які поля будуть видимими для користувачів на сторінці профілю.

- Ви можете встановити видимість поля для всіх користувачів або обмежити його лише для певних ролей.

Розташування полів на формі:

- Ви можете налаштувати порядок полів на формі реєстрації або редагування облікового запису.
- Використовуйте можливості перетягування та зміни порядку полів для досягнення бажаного розташування.

Додаткові налаштування поля:

- Кожне поле має додаткові налаштування, такі як форматування дати, валідація даних, обмеження значень тощо.
- Встановіть ці налаштування відповідно до вашої потреби та вимог.

Персоналізація макету профілю:

- Ви можете використовувати теми та шаблони Drupal для налаштування вигляду та розміщення полів на сторінці профілю користувача.

Користування полями в обліковому записі користувача дозволяє вам зберігати різноманітну інформацію про користувачів та налаштовувати її залежно від ваших потреб та вимог вашого веб-сайту.

2.5.2. КОРИСТУВАЦЬКІ РОЛІ

Ролі користувачів в Drupal 8 використовуються для групування користувачів і надання їм специфічних дозволів та привілеїв на вашому веб-сайті. Ролі дозволяють управляти доступом користувачів до різних функцій, ресурсів і вмісту на сайті.

Основні аспекти ролей користувачів в Drupal 8:

Створення ролей: У панелі адміністратора Drupal 8 ви можете створювати нові ролі користувачів. Ролі можуть мати назву і опис, що відповідають їхній функціональності або ролям у вашому проекті.

Налаштування дозволів: Кожній ролі можна привласнити різні дозволи, що визначають, які дії вона може виконувати на сайті. Дозволи контролюють доступ до різних функціональних можливостей, таких як редагування вмісту, керування модулями, налаштування сайту та інше.

Присвоєння ролей користувачам: Після створення ролей ви можете призначати їх користувачам. Користувач може мати одну або декілька ролей, що визначають його привілеї та дозволи.

Використання ролей в налаштуваннях: Ролі користувачів широко використовуються в налаштуваннях Drupal 8 для контролю доступу до функціональності та вмісту. Ви можете обмежувати доступ до окремих сторінок, блоків, виглядів та інших елементів сайту за допомогою ролей.

Переваги використання ролей користувачів в Drupal 8:

Гнучкість: Ролі дозволяють гнучко налаштовувати привілеї та дозволи для різних типів користувачів на вашому сайті. Ви можете створювати ролі відповідно до вашої специфічної структури організації або функціональних потреб.

Керованість: Ви можете легко керувати доступом користувачів, надаючи або забираючи їм ролі в залежності від їхніх потреб та ролі в проекті.

Безпека: Ролі допомагають забезпечити безпеку вашого сайту, обмежуючи доступ до конфіденційної інформації або редагування важливих налаштувань.

Недоліки використання ролей користувачів в Drupal 8:

Складність налаштування: Якщо ваш сайт має складну структуру ролей і дозволів, можуть знадобитися деякі зусилля для правильного налаштування та управління ролями.

Можливість помилок: Неправильно налаштовані ролі та дозволи можуть призвести до неправильного доступу до функцій і вмісту на сайті.

Загалом, ролі користувачів в Drupal 8 є потужним інструментом для керування доступом та привілеями користувачів на вашому веб-сайті. Їх використання дозволяє вам контролювати доступ до різних функцій та вмісту, забезпечуючи безпеку та гнучкість управління вашим сайтом.

Drupal 8 має декілька користувацьких ролей, які надаються по замовчуванню. Ось декілька основних ролей, які зазвичай присутні в новій установці Drupal 8:

Anonymous user (Анонімний користувач): Це роль, яка надається користувачам, які не авторизувалися на сайті. Вона дозволяє анонімним користувачам переглядати публічний вміст сайту.

Authenticated user (Авторизований користувач): Ця роль надається користувачам, які успішно авторизувалися на сайті. Вона дозволяє авторизованим користувачам переглядати додатковий вміст та виконувати певні дії, такі як коментування або відправка форм.

Administrator (Адміністратор): Це роль, яка надається основному адміністратору сайту. Користувач з цією роллю має повний доступ до всіх функцій та налаштувань

Drupal, включаючи управління користувачами, контентом, модулями, темами та іншими аспектами сайту.

Правильне проектування ролей для Drupal 8 допомагає забезпечити ефективне керування доступом та привілеями користувачів на вашому веб-сайті. Ось кілька кроків, які можуть допомогти вам в правильному проектуванні ролей:

Аналізуйте функціональність сайту: Детально проаналізуйте функціональні можливості вашого веб-сайту та визначте, які дії і функції повинні бути доступними для різних типів користувачів.

Ідентифікуйте типи користувачів: Визначте основні типи користувачів, які будуть використовувати ваш веб-сайт. Наприклад, це можуть бути адміністратори, редактори, авторизовані користувачі, гості тощо.

Створіть ролі на основі функціональності: Для кожного типу користувача створіть відповідну роль. Роль повинна відображати доступну функціональність та привілеї для цього типу користувача. Наприклад, адміністратор може мати доступ до всіх функцій та налаштувань, тоді як редактор може мати доступ лише до редагування вмісту.

Встановіть дозволи для ролей: Призначте відповідні дозволи для кожної ролі, щоб визначити, які дії дозволені для кожного типу користувача. Врахуйте, що дозволи можуть відрізнятися від типу контенту, модулів, настройок сайту та іншого.

Додаткові налаштування ролей: Врахуйте додаткові налаштування ролей, такі як обов'язкові поля, розташування елементів, персоналізація макету тощо.

Тестування ролей: Перевірте роботу створених ролей, протестуйте доступ та функціоналість для кожного типу користувача, щоб переконатися, що ролі налаштовані правильно.

Підтримка та оновлення: Забезпечте систематичне оновлення ролей у відповідності з розвитком вашого веб-сайту. Постійно оцінюйте потреби та змінюйте ролі відповідно до зміни вимог та структури сайту.

Важливо пам'ятати, що проектування ролей - це ітеративний процес. Ви можете вносити зміни та доповнювати ролі з часом, враховуючи зміни вимог та потреб вашого веб-сайту.

Використання ролей в Drupal 8 має свої переваги і недоліки. Ось кілька з них:

Переваги використання ролей в Drupal 8:

Гнучкість: Ролі дозволяють гнучко налаштовувати доступ та привілеї для різних типів користувачів на вашому веб-сайті. Ви можете створювати ролі відповідно до

потреб вашого проекту і привласнювати їх користувачам відповідно до їх ролі в проекті.

Безпека: Використання ролей допомагає забезпечити безпеку вашого сайту. Ви можете обмежувати доступ до конфіденційної інформації або редагування важливих налаштувань, переконуючись, що користувачі мають тільки необхідні привілеї.

Керованість: Ролі дозволяють ефективно керувати доступом користувачів до функцій та вмісту на вашому веб-сайті. Ви можете швидко встановлювати, змінювати або видаляти ролі для кожного користувача, що дозволяє забезпечити потрібні рівні доступу.

Складність коду: Використання ролей допомагає зменшити складність коду, оскільки ви можете використовувати вбудовані функції та модулі Drupal для керування доступом та привілеями. Це зменшує кількість необхідного власного коду.

Недоліки використання ролей в Drupal 8:

Складність налаштування: Якщо ваш сайт має складну структуру ролей та дозволів, можуть знадобитися деякі зусилля для правильного налаштування та управління ролями. Неправильне налаштування може призвести до недостатнього або надмірного доступу для користувачів.

Можливість помилок: При неправильному налаштуванні ролей і дозволів можуть виникати помилки або конфлікти, що можуть вплинути на роботу вашого веб-сайту. Необхідно бути уважним при налаштуванні та перевірці ролей.

Складність управління: З ростом кількості користувачів та ролей управління ними може стати складним завданням. Вам потрібно буде слідкувати за змінами, оновлювати ролі та забезпечувати відповідні рівні доступу.

Потреба в документації: Використання ролей може потребувати документації, описує правила та налаштування доступу. Це допомагає забезпечити однорідність і зрозумілість налаштувань для адміністраторів та інших відповідальних осіб.

Усупереч недолікам, використання ролей є корисним інструментом для керування доступом та привілеями на вашому веб-сайті. Правильне налаштування ролей забезпечує безпеку та ефективне управління вашим сайтом.

2.5.3. ПРАВА ДОСТУПУ

В Drupal 8 права доступу використовуються для керування тим, який користувач має доступ до яких функцій та вмісту на веб-сайті. Завдяки правам доступу можна обмежувати чи надавати можливості користувачам в залежності від їх ролі та рівня доступу.

Основні концепції та компоненти прав доступу в Drupal 8:

Роль користувача: Ролі - це групи, до яких входять користувачі. Кожному користувачеві можна призначити одну або кілька ролей. Ролі визначають набір прав доступу для користувачів.

Дозволи: Дозволи - це окремі права доступу, які визначають, які дії дозволені або заборонені для певних ролей користувачів. Дозволи включають можливості, такі як перегляд, створення, редагування або видалення вмісту, керування модулями, доступ до налаштувань тощо.

Модуль Permissions: Модуль Permissions в Drupal 8 дозволяє адміністратору налаштовувати дозволи для ролей користувачів. За допомогою цього модуля можна вибрати, які дозволи мають бути доступні для кожної ролі.

PERMISSION	ANONYMOUS USER	AUTHENTICATED USER	ADMINISTRATOR
Block			
Administer blocks	☐	☐	☑
Comment			
Administer comment types and settings *Warning: Give to trusted roles only; this permission has security implications.*	☐	☐	☑
Administer comments and comment settings	☐	☐	☑
Edit own comments	☐	☐	☑
Post comments	☐	☑	
Skip comment approval	☐	☑	
View comments	☑	☑	
Configuration Manager			
Export configuration *Warning: Give to trusted roles only; this permission has security implications.*	☐	☐	☑

Рис. 10. Управління правами доступу

Контроль доступу до вмісту: Drupal 8 також надає можливість контролювати доступ до окремих сторінок, вузлів (nodes) або інших типів вмісту на вашому веб-сайті. Ви можете встановлювати права доступу до вмісту на основі ролей користувачів, наприклад, дозволяти лише певним ролям переглядати або редагувати вміст.

Переваги використання прав доступу в Drupal 8:

Гнучкість: Ви можете точно налаштовувати права доступу для різних ролей користувачів на вашому веб-сайті.

Безпека: Правильне налаштування прав доступу допомагає забезпечити безпеку вашого сайту, обмежуючи доступ до конфіденційної інформації або важливих функцій.

Управління користувачами: Ви можете легко керувати ролями та правами доступу для користувачів, надаючи або забираючи їм певні можливості.

Недоліки використання прав доступу в Drupal 8:

Складність налаштування: При складній структурі ролей та дозволів можуть виникати певні виклики управління та налаштування прав доступу.

Потреба в уважному контролі: Неправильно налаштовані права доступу можуть призвести до недостатнього або надмірного доступу до функцій та вмісту сайту.

Потреба в систематичному оновленні: При зміні ролей, дозволів або структури сайту, необхідно вносити зміни до прав доступу та оновлювати їх у відповідності до нових вимог.

В цілому, права доступу в Drupal 8 дозволяють ефективно керувати доступом користувачів до функцій та вмісту на вашому веб-сайті, забезпечуючи безпеку та гнучкість управління.

Призначення прав доступу в Drupal 8 - це важлива частина налаштування вашого веб-сайту. Ось кілька кроків, які можуть допомогти вам правильно призначати права доступу в Drupal 8:

Аналізуйте потреби: Ретельно проаналізуйте потреби вашого веб-сайту та користувачів. Визначте, які дії та функції повинні бути доступні для різних ролей користувачів.

Визначте ролі користувачів: Виберіть основні типи користувачів на вашому веб-сайті та створіть відповідні ролі для них. Наприклад, це можуть бути адміністратори, редактори, авторизовані користувачі, гості тощо.

Налаштуйте дозволи: Призначте відповідні дозволи для кожної ролі, щоб визначити, які дії дозволені для кожного типу користувача. Використовуйте модуль Permissions для зручного налаштування дозволів. Пам'ятайте, що необхідно обмежувати доступ до критичних дій та конфіденційної інформації.

Перевірте дозволи за замовчуванням: Перевірте, які дозволи призначені за замовчуванням для ролей, щоб забезпечити безпеку та відповідність.

Тестування: Перевірте роботу призначених прав доступу, протестуйте доступ та функціональність для кожної ролі. Переконайтеся, що кожна роль має лише необхідні привілеї та доступ.

Постійне оновлення: Регулярно переглядайте і оновлюйте права доступу, особливо при зміні структури сайту, додаванні нових функцій або відповідно до зміни вимог.

Важливо слідкувати за правильним налаштуванням прав доступу, оскільки неправильно призначені права можуть призвести до недостатнього або надмірного доступу до функцій і вмісту на вашому веб-сайті.

2.5.4. ПІДХОДИ ДО ОРГАНІЗАЦІЇ РОЗПОДІЛЕНОГО ДОСТУПУ ДО КОНТЕНТУ

Друпал 8 не має вбудованих засобів для розподіленого доступу до контенту, але можна використовувати різні модулі та підходи для досягнення цієї мети. Ось декілька можливих підходів:

Модуль Organic Groups: Модуль Organic Groups дозволяє створювати групи користувачів і призначати доступ до вмісту на основі цих груп. Кожна група може мати свої власні правила доступу та може мати власних користувачів, що мають доступ до контенту в межах групи.

Модуль Domain Access: Модуль Domain Access дозволяє створювати різні домени (або піддомени) і призначати доступ до контенту на основі цих доменів. Кожен домен може мати свою власну структуру контенту і правила доступу.

Модуль Content Access: Модуль Content Access надає додаткові налаштування доступу до вмісту на основі ролей та прав користувачів. Ви можете призначати різні рівні доступу до конкретних типів вмісту для різних ролей користувачів.

Підходи на рівні коду: Якщо ви маєте складні вимоги щодо розподіленого доступу до контенту, ви можете використовувати власні налаштування на рівні коду. Ви можете створити власні модулі, які реалізують логіку доступу до контенту на основі ваших потреб і правил.

Це лише деякі підходи, які можна використовувати для розподіленого доступу до контенту в Drupal 8. Вибір конкретного підходу залежить від ваших потреб та специфіки вашого проекту. Рекомендую детально ознайомитися з документацією та спільнотою Drupal, щоб знайти найкращий підхід для вашого проекту.

Модуль Organic Groups (OG) є розширенням для Drupal 8, яке дозволяє створювати і керувати групами користувачів і призначати доступ до вмісту в межах цих груп. Organic Groups дозволяє створювати веб-сайти зі спільнотами користувачів, де

кожна група може мати свої власні члени, контент, функції та правила доступу. Ось деякі ключові функції та концепції, пов'язані з модулем Organic Groups:

Групи: Модуль Organic Groups додає новий тип сутності "Група". Ви можете створювати групи і налаштовувати їх параметри, такі як назва, опис, зображення та налаштування доступу.

Типи груп: Ви можете визначити власні типи груп для відповідності специфічним потребам вашого проекту. Наприклад, це можуть бути "Проекти", "Команди", "Курси" тощо. Кожен тип груп може мати власні налаштування і функціонал.

Ролі користувачів у групі: Користувачі можуть бути членами однієї або декількох груп. Для кожної групи можна призначати ролі, такі як "Адміністратор", "Модератор", "Учасник" тощо. Ролі визначають набір прав і доступ до функцій для кожного користувача в межах групи.

Менеджер груп: Модуль Organic Groups надає адміністраторам веб-сайту інтерфейс для керування групами. Ви можете додавати, редагувати та видаляти групи, налаштовувати їх параметри, призначати користувачів та ролі.

Доступ до контенту: Вміст може бути пов'язаний з групами, що дозволяє обмежувати доступ до вмісту лише для членів конкретної групи. Ви можете налаштовувати правила доступу до вмісту для кожної групи, визначаючи, хто може переглядати, створювати, редагувати або видаляти вміст.

Контекст групи: Organic Groups додає "контекст групи" для веб-сайту. Це означає, що на сторінках вмісту або інших елементах сайту можуть відображатись тільки групи та пов'язаний з ними контент.

Переваги використання модуля Organic Groups:

Створення спільнот: Ви можете легко створювати веб-сайти зі спільнотами користувачів, де кожна група має своїх членів, вміст і функціонал.

Гнучкість налаштувань: Модуль Organic Groups надає широкі можливості налаштувань, таких як типи груп, ролі користувачів, доступ до вмісту та інші параметри, що дозволяють створювати різноманітні спільноти з унікальними правилами та функціоналом.

Керування групами: Менеджер груп в Organic Groups дозволяє зручно керувати групами, додавати нові, редагувати і видаляти існуючі, налаштовувати права доступу та ролі.

Недоліки використання модуля Organic Groups:

Складність: Organic Groups є потужним і гнучким модулем, але налаштування його функцій можуть бути складними, особливо для новачків. Вимагається деякий рівень технічного розуміння Drupal і його архітектури.

Сумісність з іншими модулями: Не всі модулі можуть працювати належним чином з модулем Organic Groups, тому перед використанням слід перевірити сумісність з іншими розширеннями, які ви плануєте використовувати на вашому веб-сайті.

Узагальнюючи, модуль Organic Groups є потужним і корисним додатком для створення веб-сайтів зі спільнотами користувачів. Він дозволяє створювати групи, призначати ролі та налаштовувати доступ до контенту. Проте, перед використанням модуля Organic Groups слід ретельно ознайомитися з документацією, дотримуватися рекомендацій і проводити тестування для забезпечення належної роботи та сумісності з вашим проектом.

Модуль Domain Access є розширенням для Drupal 8, яке дозволяє створювати і керувати різними доменами на вашому веб-сайті і призначати доступ до контенту на основі цих доменів. Він дозволяє створювати мультидоменні веб-сайти з окремими налаштуваннями, вмістом та конфігурацією для кожного домену. Ось деякі ключові функції та концепції, пов'язані з модулем Domain Access:

Домени: Модуль Domain Access додає поняття доменів до Drupal. Ви можете визначити різні домени (наприклад, example.com, example.net) і налаштувати їх параметри, такі як назва, URL, налаштування кешування тощо.

Контекст домену: Кожен домен має свій власний контекст, що означає, що різний вміст та налаштування можуть бути використані для кожного домену. Наприклад, ви можете мати різний набір блоків, меню та тему оформлення для кожного домену.

Розділення вмісту: Модуль Domain Access дозволяє вам призначати вміст до певних доменів або до всіх доменів. Це означає, що ви можете мати загальний вміст для всіх доменів або окремий вміст для кожного домену.

Налаштування доступу: Ви можете налаштувати правила доступу до контенту на основі доменів. Наприклад, ви можете визначити, хто може переглядати, створювати, редагувати або видаляти вміст для кожного домену.

Конфігурація блоків та меню: Модуль Domain Access дозволяє налаштовувати, які блоки та меню відображаються для кожного домену. Ви можете визначити, які блоки будуть видимі або приховані на кожному домені.

Переваги використання модуля Domain Access:

Мультидоменна підтримка: Модуль Domain Access дозволяє вам створювати мультидоменні веб-сайти з окремими налаштуваннями та контентом для кожного домену.

Легкість у використанні: Модуль має зручний інтерфейс для налаштування доменів, контекстів та доступу до контенту.

Гнучкість: Ви можете налаштувати різні типи доменів з різними правилами доступу та параметрами.

Недоліки використання модуля Domain Access:

Складність налаштування: Налаштування модуля Domain Access може бути складним завданням, особливо для новачків. Рекомендується уважно ознайомитися з документацією та проводити тестування перед використанням модуля на живому веб-сайті.

Сумісність з іншими модулями: Не всі модулі можуть працювати належним чином з модулем Domain Access, тому перед використанням слід перевірити сумісність з іншими розширеннями, які ви плануєте використовувати на вашому веб-сайті.

Узагальнюючи, модуль Domain Access є корисним додатком для створення мультидоменних веб-сайтів з різними налаштуваннями, контентом та доступом. Він дозволяє керувати різними аспектами вашого веб-сайту в залежності від доменів.

Модуль Content Access є розширенням для Drupal 8, яке надає додаткові можливості для керування доступом до вмісту на вашому веб-сайті. Він дозволяє налаштовувати правила доступу на основі ролей користувачів та індивідуальних налаштувань контенту. Ось детальніше про ключові функції та концепції, пов'язані з модулем Content Access:

Правила доступу: Модуль Content Access дозволяє вам створювати правила доступу до вмісту. Ви можете налаштовувати, які ролі користувачів мають право переглядати, створювати, редагувати або видаляти вміст.

Ролі користувачів: Ви можете призначати ролі користувачів і налаштовувати їх права доступу до вмісту. Наприклад, ви можете визначити, що адміністратори мають повний доступ до всього вмісту, а редактори можуть редагувати, але не можуть видаляти вміст.

Індивідуальні налаштування контенту: Крім ролей користувачів, модуль Content Access дозволяє налаштовувати індивідуальні правила доступу до окремих вузлів або інших типів вмісту. Ви можете призначати права доступу до конкретних вмістових елементів для окремих користувачів або груп користувачів.

Наслідування прав доступу: Модуль Content Access дозволяє використовувати наслідування прав доступу. Це означає, що ви можете налаштувати правила доступу до батьківського вузла, які автоматично застосовуються до всього дочірнього вмісту.

Контроль доступу до полів: Модуль також дозволяє налаштовувати доступ до окремих полів вузлів. Ви можете встановлювати правила доступу до певних полів, забороняючи користувачам редагувати або переглядати їх.

Переваги використання модуля Content Access:

Гнучкість налаштувань: Модуль дозволяє детально налаштовувати правила доступу до вмісту на основі ролей користувачів, індивідуальних налаштувань контенту та інших факторів.

Контроль доступу до полів: Модуль дозволяє точно керувати доступом до окремих полів вузлів, що є корисним для дотримання конфіденційності даних.

Недоліки використання модуля Content Access:

Складність налаштування: Модуль може бути складним для налаштування, особливо якщо вам потрібно встановити складні правила доступу або контроль доступу до багатьох полів.

Сумісність з іншими модулями: В деяких випадках модуль Content Access може впливати на сумісність з іншими модулями, особливо тими, які також впливають на керування доступом до вмісту. Рекомендується перевіряти сумісність та тестувати налаштування перед використанням на живому веб-сайті.

Узагальнюючи, модуль Content Access є потужним інструментом для керування доступом до вмісту на веб-сайті. Він дозволяє налаштовувати правила доступу на основі ролей користувачів та індивідуальних налаштувань контенту, забезпечуючи гнучкість та контроль над доступом до вмісту на вашому сайті.

2.5.5. ЗАСОБИ ДЛЯ РОЗШИРЕННЯ ФУНКЦІОНАЛУ УПРАВЛІННЯ ДОСТУПОМ

Модуль Workbench Access є розширенням для Drupal 8, яке надає додаткові функціональні можливості для керування доступом до контенту на вашому веб-сайті, зокрема для сайтів зі складною структурою та багатьма редакторами. Ось детальніше про ключові функції та концепції, пов'язані з модулем Workbench Access:

Робочі області: Модуль Workbench Access додає поняття "робочих областей" до Drupal. Робоча область - це контекст, в якому ви працюєте з контентом. Ви можете мати окремі робочі області для різних редакторів або відділів.

Керування доступом: Модуль дозволяє налаштовувати правила доступу до робочих областей та контенту. Ви можете призначати ролі користувачів для кожної робочої області і визначати їх права доступу.

Редагування і публікація: Модуль Workbench Access дозволяє редакторам працювати зі своїм власним контентом у відповідних робочих областях. Вони можуть редагувати контент, переглядати його попередній перегляд та публікувати зміни.

Перегляд і схвалення: Модуль також надає можливість налаштування процесу схвалення контенту. Ви можете встановлювати правила, які контент потребує перегляду та схвалення перед публікацією.

Інформаційні панелі: Модуль Workbench Access надає інформаційні панелі для керівництва, які дозволяють редакторам бачити перегляди та зміни, які відбуваються в їхній робочій області.

Переваги використання модуля Workbench Access:

Гнучкість налаштувань: Модуль дозволяє гнучко налаштовувати правила доступу до контенту в залежності від потреб вашого веб-сайту та структури організації.

Контроль доступу до робочих областей: Модуль дозволяє створювати окремі робочі області для різних редакторів або відділів, забезпечуючи ізольований доступ до контенту.

Недоліки використання модуля Workbench Access:

Складність налаштування: Модуль може бути складним для налаштування, особливо якщо ви маєте складну структуру робочих областей або складні правила доступу.

Сумісність з іншими модулями: Ви повинні впевнитися, що модуль сумісний з іншими модулями, які ви використовуєте на вашому веб-сайті, особливо тими, які впливають на керування доступом та роботу з контентом.

Узагальнюючи, модуль Workbench Access є потужним інструментом для керування доступом до контенту на вашому веб-сайті, забезпечуючи гнучкість налаштувань та контроль доступу до робочих областей. Він підходить для веб-сайтів зі складною структурою, багатьма редакторами та потребою у вищому рівні контролю над доступом до контенту.

Модуль Field Access є розширенням для Drupal 8, яке надає додаткові можливості для керування доступом до поля вмісту. Він дозволяє налаштовувати правила доступу до окремих полів вузлів на основі ролей користувачів і інших

факторів. Ось детальніше про ключові функції та концепції, пов'язані з модулем Field Access:

Правила доступу до полів: Модуль Field Access дозволяє вам визначити правила доступу до окремих полів вузлів. Ви можете налаштовувати, які ролі користувачів мають право переглядати, редагувати або видаляти дані в цих полях.

Керування доступом на основі ролей: Модуль дозволяє вам призначати ролі користувачів і налаштовувати їх права доступу до полів. Наприклад, ви можете визначити, що тільки адміністратори мають право редагувати певне поле, а решта користувачів можуть тільки переглядати його значення.

Контроль доступу до конкретних значень: Модуль також дозволяє вам налаштовувати доступ до конкретних значень полів для окремих користувачів. Ви можете встановлювати, хто має право переглядати, редагувати або видаляти певні значення полів.

Налаштування доступу на основі контексту: Модуль Field Access дозволяє налаштовувати доступ до полів на основі контексту, такого як тип вмісту, стан публікації, власник вузла та інші фактори.

Переваги використання модуля Field Access:

Гнучкість управління доступом: Модуль дозволяє детально налаштовувати правила доступу до полів вузлів на основі потреб вашого веб-сайту.

Забезпечення конфіденційності даних: Модуль Field Access дозволяє контролювати, хто має право переглядати, редагувати або видаляти певні дані в полях, що допомагає забезпечити конфіденційність даних.

Недоліки використання модуля Field Access:

Складність налаштування: Модуль може бути складним для налаштування, особливо якщо ви маєте складні правила доступу до багатьох полів або різних типів вмісту.

Сумісність з іншими модулями: При використанні модуля Field Access слід перевірити його сумісність з іншими модулями, які ви використовуєте на вашому веб-сайті, особливо з модулями, що впливають на керування доступом або роботу з полями.

Узагальнюючи, модуль Field Access є корисним інструментом для керування доступом до полів вузлів в Drupal 8. Він дозволяє гнучко налаштовувати правила доступу на основі ролей користувачів і інших факторів, забезпечуючи більшу контроль над доступом до даних у вашому веб-сайті.

2.5.6. ВИКОРИСТАННЯ ОБЛІКОВИХ ЗАПИСІВ ТА РОЛЕЙ ДЛЯ УПРАВЛІННЯ ДОСТУПОМ ДО КОНТЕНТУ

Управління доступом до контенту в Drupal 8 може здійснюватися різними способами, в залежності від потреб вашого веб-сайту. Ось декілька типових підходів до управління доступом до контенту в Drupal 8:

Використання ролей користувачів: В Drupal 8 ви можете використовувати вбудований механізм ролей користувачів для призначення прав доступу до контенту. Ви можете створювати різні ролі для користувачів і надавати їм відповідні права доступу до контенту, використовуючи модуль "Ролі і дозволи". Цей підхід дозволяє легко управляти доступом на рівні ролей.

Використання модулів керування доступом: Drupal 8 також надає ряд модулів, які допомагають розширити можливості управління доступом. Наприклад, модуль "Content Access" дозволяє налаштовувати правила доступу до окремих вузлів, а модуль "Field Permissions" дозволяє налаштовувати доступ до окремих полів вузлів. Використання таких модулів дозволяє більш детально контролювати доступ до контенту на основі потреб вашого веб-сайту.

Використання модулів груп та облікових записів: Якщо ви маєте веб-сайт зі складною структурою або багатьма користувачами, ви можете використовувати модулі, які дозволяють організовувати контент у групи або облікові записи. Наприклад, модуль "Organic Groups" дозволяє створювати групи користувачів та обмежувати доступ до контенту на рівні груп. Цей підхід корисний для сайтів, де різні групи користувачів повинні мати різний доступ до контенту.

Типові помилки при управлінні доступом до контенту в Drupal 8:

Недостатньо докладне налаштування прав доступу: Одна з типових помилок полягає у недостатньому докладному налаштуванні прав доступу. Упевніться, що ви ретельно налаштували правила доступу до контенту відповідно до потреб вашого веб-сайту. Перевірте права доступу для ролей користувачів і впевніться, що вони відповідають вашим очікуванням.

Надмірне навантаження прав доступу: Іноді може виникати проблема з надмірним навантаженням на сервер через складність прав доступу. Важливо забезпечити, щоб налаштування прав доступу було ефективним та оптимізованим, щоб уникнути зайвого навантаження на сервер.

Несумісність з іншими модулями: Іноді використання декількох модулів для управління доступом може викликати проблеми сумісності. Перед встановленням та

налаштуванням модулів переконайтеся, що вони сумісні між собою та з іншими модулями, які ви використовуєте на вашому веб-сайті.

Управління доступом до контенту - важлива частина будь-якого веб-сайту, і Drupal 8 надає ряд потужних інструментів для забезпечення контролю та безпеки вашого контенту. Важливо ретельно розглянути ваші потреби та вибрати відповідні модулі та стратегії управління доступом, щоб забезпечити ефективне та безпечне функціонування вашого веб-сайту.

Питання для самоконтролю:

1. Що таке ролі?
2. Як налаштовуються права доступу?
3. Які ролі створені по замовченню?
4. Які модулі використовуються для розширення типового функціоналу керування доступом?
5. Які типові помилки керування доступом?

2.6. УПРАВЛІННЯ КОНФІГУРАЦІЯМИ В DRUPAL

Конфігурація в Drupal 8 використовується для збереження налаштувань вашого веб-сайту, таких як модулі, теми, блоки, фільтри, правила доступу та інші параметри. Ось детальніше про конфігурації в Drupal 8:

Конфігураційні об'єкти: В Drupal 8 конфігурація зберігається у вигляді конфігураційних об'єктів, які мають структуровану форму. Кожен модуль або функціонал Drupal може мати свої власні конфігураційні об'єкти, які зберігаються у вигляді YML-файлів.

Файли конфігурації: Конфігураційні файли зберігаються у папці "config" вашого Drupal-сайту. Кожен модуль має свою власну підпапку в "config" для збереження конфігураційних файлів. Конфігураційні файли мають розширення ".yml" і містять ключі та значення для налаштувань.

Управління конфігурацією: Drupal 8 надає інтерфейс управління конфігурацією, який дозволяє змінювати та зберігати конфігураційні параметри. Інтерфейс управління конфігурацією дозволяє вам переглядати налаштування, редагувати їх і зберігати зміни.

Імпорт та експорт конфігурації: Ви можете імпортувати та експортувати конфігурацію в Drupal 8. Це дозволяє вам легко переносити налаштування між різними середовищами розробки, наприклад, з локального сервера на продакшн.

Друштування конфігурації: В Drupal 8 ви можете використовувати систему друштування конфігурації для керування версіями конфігураційних файлів. Це дозволяє вам контролювати та відстежувати зміни в конфігурації вашого веб-сайту.

Переваги використання конфігурацій в Drupal 8:

Зручне управління налаштуваннями: Конфігурація в Drupal 8 дозволяє зручно керувати налаштуваннями вашого веб-сайту через інтерфейс управління конфігурацією.

Легка міграція та розгортання: Завдяки можливості імпорту та експорту конфігурації, ви можете легко мігрувати налаштування між середовищами розробки та розгортати їх на продакшн-сервер.

Недоліки використання конфігурацій в Drupal 8:

Складність при налаштуванні: Деякі аспекти конфігурації можуть бути складними для налаштування, особливо при використанні складних модулів або кастомних налаштувань.

Помилки в конфігураційних файлах: Неправильні зміни в конфігураційних файлах можуть призвести до помилок або некоректної роботи вашого веб-сайту. Тому важливо бути уважним і перевіряти зміни перед збереженням.

Узагальнюючи, конфігурація в Drupal 8 дозволяє зберігати та керувати налаштуваннями вашого веб-сайту. Це зручний інструмент, який допомагає вам легко налаштувати ваш веб-сайт та ефективно управляти його параметрами.

2.6.1. ПОНЯТТЯ КОНФІГУРАЦІЇ

Конфігурація в Drupal 8 відноситься до збереження налаштувань вашого веб-сайту, які використовуються для налаштування модулів, тем, блоків, фільтрів, прав доступу та багатьох інших параметрів. Ось деякі ключові поняття, пов'язані з конфігурацією в Drupal 8:

Конфігураційні об'єкти: Конфігурація в Drupal 8 представлена у вигляді конфігураційних об'єктів, які мають структурований формат. Кожен модуль або функціонал Drupal може мати свої власні конфігураційні об'єкти, які зберігаються у вигляді файлів формату YAML (YAML Ain't Markup Language).

Конфігураційні файли: Конфігураційні файли містять налаштування для різних аспектів вашого веб-сайту. Вони зберігаються у папці "config" вашого Drupal-сайту і мають розширення ".yml". Кожен модуль може мати свою власну підпапку в "config" для збереження конфігураційних файлів.

Інтерфейс управління конфігурацією: Drupal 8 надає інтерфейс управління конфігурацією, який дозволяє адміністраторам вносити зміни в налаштування веб-сайту. Цей інтерфейс дозволяє переглядати, редагувати та зберігати конфігураційні параметри.

Механізм імпорту та експорту: Drupal 8 має вбудовану систему імпорту та експорту конфігурації, яка дозволяє переносити налаштування між різними середовищами розробки або розгортати їх на продакшн-сервер. Це забезпечує простоту управління налаштуваннями та дозволяє легко розгорнути ваш веб-сайт на різних серверах.

Шаблонізація конфігурації: Drupal 8 також підтримує можливість шаблонізації конфігурації, що дозволяє вам створювати шаблонні файли конфігурації для використання на різних серверах або в різних середовищах розробки.

Використання конфігурації в Drupal 8 дозволяє зручно керувати налаштуваннями вашого веб-сайту, зберігаючи їх у структурованому форматі. Це спрощує процес налаштування, управління та розгортання вашого веб-сайту.

Різниця між конфігураціями та State API в Drupal 8 є ключовою для розуміння, які дані слід зберігати в кожному з цих механізмів. Давайте розглянемо цю різницю докладніше:

Конфігурації:

- Конфігурація використовується для збереження налаштувань вашого веб-сайту, які можуть змінювати адміністратори.
- Це статичні дані, які не змінюються протягом одного запиту.
- Налаштування модулів, тем, блоків, фільтрів і багатьох інших параметрів зберігаються у вигляді конфігураційних об'єктів.
- Конфігурація зберігається у вигляді YAML-файлів у папці "config" вашого Drupal-сайту.
- Конфігурація може бути імпортована та експортована між різними середовищами розробки і розгортання.

State API:

- State API використовується для збереження тимчасових або незмінних даних, які використовуються на протязі одного запиту або для збереження стану вашого веб-сайту.
- Це дані, які можуть змінюватися протягом одного запиту і не потребують довгострокового збереження.
- Поточна мова, налаштування підключеного модуля, останній переглянутий запис - приклади даних, які можуть бути збережені за допомогою State API.
- Дані State API зберігаються в базі даних Drupal і доступні для використання на протязі одного запиту.
- State API не надає можливості імпорту та експорту, оскільки він залежить від поточного стану веб-сайту.

Отже, конфігурації використовуються для збереження статичних налаштувань вашого веб-сайту, які можуть змінювати адміністратори, в той час як State API використовується для збереження тимчасових або незмінних даних, які використовуються на протязі одного запиту або для збереження стану вашого

веб-сайту. Коректне розуміння різниці між цими двома механізмами допоможе вам ефективно використовувати їх у своєму проекті Drupal 8.

2.6.2. МЕНЕДЖЕР КОНФІГУРАЦІЙ ТА ЙОГО ВИКОРИСТАННЯ

Configuration Manager (Менеджер конфігурації) є одним з основних модулів Drupal 8, який забезпечує керування конфігурацією вашого веб-сайту. Він надає зручний інтерфейс для імпорту, експорту, збереження та відновлення конфігураційних параметрів, що дозволяє ефективно керувати налаштуваннями вашого сайту. Ось деталізований огляд Configuration Manager:

Збереження конфігурації:

- Configuration Manager дозволяє зберігати конфігурацію вашого веб-сайту у вигляді конфігураційних об'єктів.
- Конфігурація зберігається у вигляді YAML-файлів в папці "config" вашого Drupal-сайту.
- Конфігураційні файли включають налаштування модулів, тем, блоків, фільтрів, прав доступу та багатьох інших аспектів вашого веб-сайту.

Імпорт та експорт конфігурації:

- Configuration Manager дозволяє імпортувати та експортувати конфігурацію між різними середовищами розробки та розгортання.
- Імпорт дозволяє вам внести зміни в конфігурацію зовнішніми файлами YAML, що дозволяє легко переміщувати налаштування між серверами.
- Експорт генерує YAML-файли на основі поточного стану конфігурації вашого сайту.

Управління версіями:

- Configuration Manager дозволяє вам відстежувати та контролювати зміни в конфігурації шляхом використання систем контролю версій, таких як Git.
- Зміни в конфігураційних файлах можуть бути легко відстежені та відновлені, що забезпечує контроль історії змін.

Управління залежностями:

- Configuration Manager дозволяє вам керувати залежностями між конфігураційними об'єктами, що дозволяє контролювати порядок імпорту та експорту.

- Ви можете вказати, які об'єкти повинні бути імпортовані або експортовані спочатку, забезпечуючи правильне налаштування вашого веб-сайту.

Інтерфейс користувача:

- Configuration Manager має зручний інтерфейс управління, що дозволяє адміністраторам переглядати, редагувати, видаляти та зберігати конфігураційні параметри.
- Інтерфейс надає також можливість перевіряти стан конфігурації, виявляти розбіжності між конфігураційними файлами та поточним станом сайту.

Configuration Manager є потужним інструментом управління конфігурацією вашого веб-сайту в Drupal 8. Він допомагає ефективно керувати налаштуваннями та забезпечує зручний спосіб імпорту та експорту конфігурації для полегшення розробки та розгортання вашого сайту.

Використання менеджера конфігурацій (Configuration Manager) в Drupal 8 має свої переваги і недоліки, які варто враховувати при розробці та управлінні вашим веб-сайтом. Ось кілька ключових переваг і недоліків:

Переваги:

Легкість імпорту та експорту: Менеджер конфігурацій дозволяє легко імпортувати та експортувати конфігурацію між різними середовищами розробки і розгортання. Це спрощує процес розгортання та дозволяє легко переміщувати налаштування між серверами.

Збереження конфігурації у вигляді файлів: Конфігураційні параметри зберігаються у вигляді YAML-файлів, що дозволяє зручно керувати налаштуваннями вашого веб-сайту. Це спрощує контроль версій і забезпечує зручний інтерфейс для перегляду та редагування конфігураційних файлів.

Зручне керування залежностями: Configuration Manager дозволяє керувати залежностями між конфігураційними об'єктами. Ви можете вказати порядок імпорту та експорту, забезпечуючи правильне налаштування вашого веб-сайту.

Система контролю версій: Configuration Manager інтегрується з системами контролю версій, такими як Git, що дозволяє вам відстежувати та контролювати зміни в конфігурації вашого веб-сайту. Це полегшує спільну роботу над конфігурацією та відновлення попередніх версій.

Недоліки:

Вимоги до налагодження: Використання менеджера конфігурацій потребує певних налагоджень, налаштування дозволів файлів і правильного розміщення конфігураційних файлів. Неправильне налаштування може призвести до проблем з імпортом та експортом конфігурації.

Незручність для динамічних змін: Конфігураційні параметри в Drupal 8 є статичними і не підтримують динамічні зміни в режимі реального часу. Якщо вам потрібно внести швидкі зміни в конфігурацію, вам може знадобитись перезавантаження кешу або внесення змін безпосередньо до бази даних.

Великі об'єми конфігураційних файлів: Зі зростанням об'єму налаштувань вашого веб-сайту, конфігураційні файли можуть стати великими і складними для керування. Це може призвести до збільшення часу імпорту та експорту конфігурації.

Ризик конфліктів: Якщо два або більше адміністраторів одночасно вносять зміни в конфігурацію і запускають імпорт або експорт, може виникнути конфлікт, що призведе до проблем зі збереженням змін.

Незважаючи на ці недоліки, менеджер конфігурацій є потужним інструментом для керування конфігурацією в Drupal 8. Він полегшує розробку, розгортання та управління конфігурацією вашого веб-сайту.

Користування менеджером конфігурацій (Configuration Manager) в Drupal 8 дозволяє вам легко керувати налаштуваннями вашого веб-сайту. Цей інструмент дозволяє імпортувати та експортувати конфігураційні файли, зберігати їх у вигляді YAML-файлів та контролювати залежності між конфігураційними об'єктами. Ось докладний опис, як користуватись менеджером конфігурацій:

Імпорт конфігурації:

- Перейдіть на сторінку "/admin/config/development/configuration".
- Натисніть кнопку "Import".
- Виберіть YAML-файли, які містять конфігурацію для імпорту.
- Натисніть кнопку "Import" для імпорту конфігурації.
- Ви отримаєте повідомлення про успішний імпорт конфігурації.

Експорт конфігурації:

- Перейдіть на сторінку "/admin/config/development/configuration".
- Виберіть об'єкти конфігурації, які ви хочете експортувати.
- Натисніть кнопку "Export".
- Конфігураційні файли будуть згенеровані у вигляді YAML-файлів і доступні для завантаження.

Керування конфігурацією:

- На сторінці "/admin/config/development/configuration" ви можете переглянути всі конфігураційні об'єкти.
- Щоб редагувати конфігурацію, натисніть кнопку "Edit" поряд з об'єктом, який ви хочете змінити.
- Зробіть необхідні зміни у формі редагування конфігурації.
- Натисніть кнопку "Save" для збереження змін.

Контроль версій:

- Конфігураційні файли можна керувати за допомогою системи контролю версій, наприклад Git.
- Зберігайте свої YAML-файли в репозиторії контролю версій, щоб зберегти історію змін та легко відстежувати різні версії конфігурації.

Користування менеджером конфігурацій дозволяє вам зручно керувати налаштуваннями вашого веб-сайту, забезпечує зручну інтерфейс для імпорту та експорту конфігурації і спрощує роботу з командами управління конфігурацією. Пам'ятайте про важливість збереження резервних копій конфігураційних файлів та обережне внесення змін для уникнення можливих проблем.

2.6.3. ЕКСПОРТ КОНФІГУРАЦІЙ

Експорт конфігурацій в Drupal 8 дозволяє вам зберігати налаштування вашого веб-сайту у вигляді YAML-файлів, які можна зберігати у системі контролю версій і легко відновлювати або переміщувати між середовищами. Ось кілька важливих моментів, на які варто звернути увагу при експорті конфігурацій:

Сторінка експорту:

- Для експорту конфігурацій перейдіть на сторінку "/admin/config/development/configuration".
- На цій сторінці ви побачите список налаштувань, які можна експортувати.

Вибір конфігураційних об'єктів:

Оберіть об'єкти конфігурації, які ви хочете експортувати.

Ви можете вибрати окремі об'єкти або вибрати всі, використовуючи прапорець "Select all".

Експорт у YAML-файли:

- Після вибору об'єктів натисніть кнопку "Export".

- Конфігураційні файли будуть згенеровані у вигляді YAML-файлів і доступні для завантаження.
- Файли будуть збережені у папку "config" вашого Drupal-сайту.

Залежності між конфігураційними об'єктами:

- Враховуйте залежності між конфігураційними об'єктами. Наприклад, якщо ви експортуєте налаштування панелі, переконайтеся, що ви також експортуєте всі залежності, такі як блоки, відображення тощо.

Перевірка вмісту YAML-файлів:

- Перед завантаженням YAML-файлів у систему контролю версій або переміщенням їх на інший сервер, перевірте їх вміст.
- Впевніться, що всі налаштування, які ви очікуєте, присутні у файлах і збережені правильно.

Резервне копіювання:

- Регулярно робіть резервні копії конфігураційних файлів, особливо перед внесенням великих змін.
- Це допоможе вам легко відновити попередню версію конфігурації у випадку проблем.

Зберігання конфігурації у вигляді YAML-файлів і правильне управління ними дозволяє забезпечити контроль версій, легкість переміщення між середовищами та зручну систему управління налаштуваннями вашого Drupal-сайту.

2.6.4. ІМПОРТ КОНФІГУРАЦІЙ

Імпорт конфігурацій в Drupal 8 дозволяє вам встановлювати налаштування для вашого веб-сайту з використанням YAML-файлів. Ось кілька важливих моментів, на які варто звернути увагу при імпорті конфігурацій:

Перед імпортом:

- Переконайтеся, що ви маєте належні права доступу до файлів конфігурації.
- Переконайтеся, що ви маєте резервні копії поточної конфігурації, на випадок якщо щось піде не так.

Формат YAML:

- Переконайтеся, що ваші YAML-файли мають правильний синтаксис.
- Будьте уважні до правильного форматування, відступів та роздільників.

Перевірка залежностей:

- Якщо ваші конфігураційні файли мають залежності від інших об'єктів, переконайтеся, що ви експортували і імпортували всі необхідні залежності.

Дублювання та конфлікти:

- При імпорті конфігурації переконайтеся, що ви не дублюєте конфігураційні об'єкти, які вже існують.
- У разі виникнення конфліктів збереження буде відмінено, і ви отримаєте повідомлення про конфлікт.

Перевірка імпорту:

- Після завершення імпорту перевірте свій веб-сайт, щоб впевнитися, що налаштування були успішно застосовані.

Загальний підхід до імпорту конфігурацій в Drupal 8 - це ретельна підготовка файлів конфігурації, перевірка залежностей та уважне слідкування за процесом імпорту. Завжди робіть резервні копії перед виконанням змін у конфігурації, щоб мати можливість легко відновити попередній стан в разі проблем.

2.6.5. ВИКОРИСТАННЯ DRUSH ДЛЯ РОБОТИ З КОНФІГУРАЦІЯМИ

Drush (Drupal Shell) є інструментом командного рядка, який надає широкий набір функцій для управління та автоматизації Drupal-сайтами. При роботі з конфігураціями в Drupal 8 Drush може бути дуже корисним. Ось декілька способів використання Drush для роботи з конфігураціями:

Експорт конфігурації:

- За допомогою команди `drush config-export` ви можете експортувати конфігурацію вашого Drupal-сайту в YAML-файли.
- Наприклад, `drush config-export` експортує всю конфігурацію, а `drush config-export views.view.example` експортує конфігурацію лише для певного об'єкта.

Імпорт конфігурації:

- За допомогою команди `drush config-import` ви можете імпортувати конфігурацію з YAML-файлів до вашого Drupal-сайту.
- Наприклад, `drush config-import` імпортує всю конфігурацію, а `drush config-import views.view.example` імпортує конфігурацію лише для певного об'єкта.

Керування конфігурацією:

- Drush надає команди для отримання інформації про налаштування, видалення конфігураційних об'єктів і перегляду значень конфігураційних ключів.
- Наприклад, `drush config-get views.view.example` показує значення конфігураційного ключа для певного об'єкта.

Керування версіями конфігурації:

Drush дозволяє вам керувати версіями конфігурації за допомогою команд `drush config-status`, `drush config-diff`, `drush config-rollback` тощо.

Наприклад, `drush config-status` показує статус змін у конфігурації, а `drush config-diff` виводить різницю між поточним і експортованим станом конфігурації.

Використання Drush для роботи з конфігураціями в Drupal 8 дозволяє вам швидко та зручно експортувати, імпортувати, керувати та версіонувати вашу конфігурацію. Крім цього, Drush надає багато інших корисних команд для роботи з Drupal, що робить його потужним інструментом при розробці та управлінні Drupal-сайтами.

2.6.6. ТИПОВІ РІШЕННЯ ДЛЯ УПРАВЛІННЯ КОНФІГУРАЦІЯМИ ПРИ РОЗРОБЦІ ВЕБ-САЙТІВ ТА ВЕБ-ДОДАТКІВ

Config Split є модулем для Drupal 8, який дозволяє розділити конфігурацію сайту на окремі набори для різних середовищ (наприклад, розробка, стейджинг, продакшн) і управляти цими наборами залежно від поточного середовища.

Основна ідея Config Split полягає в тому, що ви можете створювати окремі групи конфігураційних об'єктів для кожного середовища, а потім активувати лише ті, які потрібні для поточного середовища. Це дозволяє вам налаштовувати сайт окремо для кожного середовища без необхідності вручну видаляти або змінювати конфігурацію під час переключення між середовищами.

Основні кроки використання Config Split:

1. Встановіть та активуйте модуль Config Split на своєму Drupal-сайті.
2. Створіть окремі Split для кожного середовища, яке ви хочете налаштувати. Наприклад, розробка, стейджинг, продакшн.
3. Додайте конфігураційні об'єкти до кожного Split відповідно до потреб вашого середовища. Наприклад, включення модулів, налаштування теми, маппінг шляхів до файлової системи тощо.

4. Налаштуйте правила активації для кожного Split, вказавши, який Split повинен бути активний для кожного середовища.
5. Переконайтеся, що ви налаштовуєте gitignore або інші механізми контролю версій так, щоб конфігураційні файли для кожного Split не включалися в репозиторій або інше середовище.
6. Виконайте імпорт та експорт конфігурації, щоб застосувати зміни і забезпечити синхронізацію між середовищами.

Переваги використання Config Split:

- Зручне керування конфігурацією для різних середовищ.
- Уникнення небажаних змін в конфігурації під час перемикання середовищ.
- Забезпечення консистентності та стабільності конфігурації на різних середовищах.
- Легкість установки та налаштування.

Недоліки використання Config Split:

- Додаткова складність налаштування та управління конфігурацією.
- Потреба уважного контролю версій та керування змінами в конфігурації.
- Можливість виникнення конфліктів та непередбачених проблем під час активації конфігураційних об'єктів.

Config Split є потужним інструментом для роботи з конфігурацією в Drupal 8, зокрема при налаштуванні різних середовищ. Він дозволяє зручно та ефективно керувати конфігурацією вашого сайту на різних етапах розробки та впровадження.

Config Ignore є модулем для Drupal 8, який дозволяє вам ігнорувати певні конфігураційні об'єкти при експорті та імпорті конфігурації. Це корисний інструмент для керування конфігурацією та уникнення небажаних змін в окремих конфігураційних об'єктах. Ось кілька важливих моментів щодо використання Config Ignore:

Встановлення та активація:

Спочатку вам потрібно встановити та активувати модуль Config Ignore на вашому Drupal-сайті.

Це можна зробити через адміністративний інтерфейс Drupal або за допомогою Drush команди `drush en config_ignore`.

Конфігурація файлу .config_ignore.yml:

- Після активації модулю Config Ignore, створіть файл .config_ignore.yml у кореневій директорії вашого Drupal-сайту (поруч з файлом settings.php).

- У файлі .config_ignore.yml ви можете вказати конфігураційні об'єкти, які потрібно ігнорувати.
- Наприклад, якщо ви хочете ігнорувати конфігурацію для модуля my_module, додайте наступний рядок до файлу: `module.my_module`.

Імпорт та експорт конфігурації:

- Після налаштування .config_ignore.yml, при імпорті або експорті конфігурації, модуль Config Ignore буде ігнорувати конфігураційні об'єкти, вказані у файлі.
- Це означає, що конфігураційні об'єкти, які ви вказали у .config_ignore.yml, не будуть експортовані або імпортовані, і будуть проігноровані.

Керування змінами:

- Важливо пам'ятати, що ігноровані конфігураційні об'єкти не будуть оновлюватись при імпорті конфігурації.
- Якщо ви змінили ігноровані конфігураційні об'єкти на вашому сайті, вони не будуть синхронізовані з експортованою конфігурацією.

Управління версіями:

- При роботі з контролем версій (наприклад, Git), вам потрібно додати файл .config_ignore.yml до відстежуваного списку (tracked list) і включити його у репозиторій.
- Це важливо, щоб інші розробники, які співпрацюють над проектом, також знавали конфігураційні об'єкти, які повинні бути ігноровані.

Використання Config Ignore дозволяє керувати конфігурацією вашого Drupal-сайту більш гнучко і контролювати, які об'єкти потрібно включати або виключати з процесу імпорту та експорту конфігурації.

Питання для самоконтролю:

Що таке конфігурації?

Як імпортувати та експортувати конфігурації?

Що таке менеджер конфігурацій?

Як керувати версіями конфігурацій?

2.7. РІШЕННЯ ДЛЯ БЕЗПЕКИ ТА ЗАХИСТУ ДАНИХ В DRUPAL

Безпека даних є критично важливим аспектом веб-розробки, і Drupal 8 має вбудовані функції та рекомендації для забезпечення безпеки вашого сайту. Ось детальніше про безпеку даних в Drupal 8:

Захист від SQL-ін'єкцій: Drupal 8 використовує підготовлені запити із параметрами для запобігання SQL-ін'єкціям. Це дозволяє автоматично екранувати вхідні дані, що допомагає запобігти несанкціонованому доступу до бази даних.

Захист від XSS-атак: Drupal 8 використовує вбудовані механізми екранування, щоб запобігти XSS-атакам (Cross-Site Scripting). Всі вхідні дані автоматично екрануються перед відображенням на сторінках.

Захист від CSRF-атак: Drupal 8 має вбудовані заходи безпеки проти атак CSRF (Cross-Site Request Forgery). Всі форми мають валідний токен безпеки, що дозволяє перевіряти, чи був запит ініційований з допустимого джерела.

Автентифікація та авторизація: Drupal 8 надає механізми автентифікації користувачів, включаючи вбудовану систему управління ролями та правами доступу. Ви можете налаштовувати доступ до різних розділів та функціональності сайту залежно від ролі користувача.

Захист паролів: Drupal 8 використовує хешування паролів з солюванням для забезпечення безпеки користувацьких паролів. Це означає, що паролі зберігаються у захищеному форматі і не можуть бути легко розшифровані.

Захист файлів: Drupal 8 надає можливості обмеження доступу до завантажених файлів, щоб забезпечити контроль над доступом до конфіденційної інформації.

Оновлення безпеки: Drupal 8 активно оновлюється та підтримується командою розробників, яка виправляє виявлені безпекові проблеми та надає патчі для їх усунення. Важливо регулярно оновлювати Drupal 8 та використовувати офіційні ресурси для отримання оновлень та порад з безпеки.

Враховуючи ці особливості безпеки, важливо також розуміти, що безпека даних в Drupal 8 також залежить від правильної конфігурації сервера, використання безпечних паролів, регулярного аудиту безпеки та інших кращих практик безпеки.

Вчасні оновлення Drupal 8 є критично важливими для забезпечення безпеки та стабільності вашого веб-сайту. Ось декілька причин, чому важливо вчасно оновлювати Drupal 8:

Безпека: Оновлення Drupal 8 включають в себе виправлення вразливостей безпеки, які виявлені і підтверджені командою розробників Drupal. Це може включати

виправлення потенційних вразливостей, таких як XSS-атаки, SQL-ін'єкції, CSRF-атаки та інші. Вчасні оновлення допомагають уникнути зловживань, витоку даних та втрати контролю над вашим сайтом.

Виправлення помилок: Оновлення Drupal 8 також включають в себе виправлення помилок та дефектів програмного забезпечення. Це допомагає покращити функціональність сайту, запобігає непередбаченим помилкам та забезпечує більш стабільну роботу вашого веб-сайту.

Нові функції та покращення: Оновлення Drupal 8 можуть включати нові функції, покращення продуктивності, удосконалення користувацького інтерфейсу та інші покращення. Це дозволяє вам отримати доступ до оновлених можливостей та збільшити ефективність вашого веб-сайту.

Сумісність: Вчасні оновлення Drupal 8 допомагають забезпечити сумісність вашого сайту з іншими модулями, темами та додатками. Якщо ви не оновлюєте Drupal 8, може виникнути конфлікт з новими версіями модулів або тем, що може призвести до непередбачених проблем та недоступності функціоналу.

Спільнота та підтримка: Drupal має велику спільноту розробників, яка постійно працює над покращенням та підтримкою платформи. Вчасні оновлення допомагають забезпечити продовження підтримки вашого сайту та доступ до нових функцій, розробок та важливих ресурсів.

Враховуючи ці фактори, рекомендується регулярно перевіряти наявність оновлень Drupal 8 і вчасно їх встановлювати. Це допоможе забезпечити безпеку, стабільність та продуктивність вашого веб-сайту.

2.7.1. ЗАГАЛЬНІ ПРАВИЛА ЩОДО ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ І ЗАХИСТУ ДАНИХ

Для забезпечення безпеки даних в Drupal 8, особливо конфіденційної та чутливої інформації, варто дотримуватись наступних рекомендацій:

Оновлюйте Drupal та модулі: Регулярно оновлюйте вашу Drupal-установку та всі встановлені модулі до останніх версій. Це дозволить отримати оновлення безпеки та виправити вразливості, які можуть бути використані зловмисниками.

Встановлюйте сильні паролі: Використовуйте сильні паролі для всіх облікових записів, включаючи адміністраторські облікові записи. Сильні паролі повинні містити комбінацію великих і малих літер, цифр та спеціальних символів. Варто також активувати політику складності паролів.

Обмежуйте доступ: Налаштуйте права доступу до адміністративної панелі та інших чутливих розділів вашого сайту. Забезпечте доступ тільки потрібним користувачам та ролям, щоб уникнути несанкціонованого доступу до важливих даних.

Захистіть файлову систему: Переконайтеся, що файлова система Drupal (зокрема, папки з завантаженими файлами) належним чином захищена. Встановіть відповідні права доступу до файлів і директорій, щоб уникнути несанкціонованого доступу до завантажених файлів.

Використовуйте HTTPS: Встановіть SSL-сертифікат на вашому сайті та налаштуйте HTTPS-з'єднання для захищеного передавання даних. Це особливо важливо для передачі конфіденційної інформації, такої як паролі або особисті дані користувачів.

Використовуйте модулі безпеки: Drupal 8 має широкий вибір модулів безпеки, які допоможуть підвищити рівень безпеки вашого сайту. Наприклад, "Security Kit", "Login Security", "Two-Factor Authentication" та багато інших. Вибирайте модулі, які найкраще відповідають вашим потребам та встановлюйте їх для підвищення безпеки вашого сайту.

Моніторинг та аудит безпеки: Регулярно моніторуйте ваш сайт на ознаки вторгнень та некоректної активності. Встановіть систему аудиту безпеки, яка буде вести журнал подій та сповіщати вас про підозрілу або небезпечну активність.

Зберігайте резервні копії: Регулярно робіть резервні копії вашого сайту та бази даних. Зберігання резервних копій дозволить відновити сайт у випадку втрати даних або зламу.

Загалом, безпека даних в Drupal 8 є постійним процесом, і важливо вдосконалювати та оновлювати ваші заходи безпеки відповідно до змінюваних загроз та ризиків.

Аудит безпеки в Drupal 8 допомагає виявити потенційні вразливості та недоліки в безпеці вашого сайту. Ось кілька кроків, які можна виконати під час проведення аудиту безпеки в Drupal 8:

Перевірка версії Drupal: Переконайтеся, що ви використовуєте останню версію Drupal 8. Перевірте, чи є доступні оновлення і встановіть їх для виправлення виявлених проблем безпеки.

Огляд налаштувань безпеки: Перевірте налаштування безпеки вашого сайту в адміністративній панелі Drupal. Переконайтеся, що налаштування автентифікації,

контролю доступу, обмеження спроб входу та інші параметри відповідають найкращим практикам безпеки.

Огляд модулів: Перевірте всі встановлені модулі на наявність оновлень та вразливостей безпеки. Слідкуйте за офіційними джерелами інформації про безпеку Drupal та модулів, і вчасно оновлюйте їх.

Перевірка конфігураційних файлів: Перевірте конфігураційні файли Drupal (наприклад, settings.php) на наявність недоліків безпеки. Переконайтеся, що файли мають правильні права доступу та немістять неправильні налаштування.

Сканування вразливостей: Використовуйте спеціалізовані інструменти сканування вразливостей для Drupal, які допоможуть виявити потенційні проблеми безпеки. Наприклад, Drupalgeddon, Droopescan, OWASP Drupal Security Scanner та інші.

Аудит коду та тем: Проведіть аудит коду вашого сайту, включаючи модулі та власний код. Перевірте наявність потенційно небезпечних функцій, вразливостей XSS, SQL-ін'єкцій, некоректної обробки введення користувачів та інші проблеми безпеки.

Аналіз журналів подій: Перегляньте журнали подій вашого сайту, включаючи журнали входу, журнали сервера та журнали безпеки. Шукайте підозрілу або несанкціоновану активність, а також помилки безпеки.

Тестування на проникнення: Виконайте тестування на проникнення (pentesting) для вашого сайту, щоб виявити можливі слабкі місця та проблеми безпеки. Це може бути виконано вручну або за допомогою спеціалізованих інструментів.

Заходи безпеки сервера: Перевірте налаштування вашого сервера, включаючи файрвол, SSL-сертифікати, налаштування безпеки HTTP та інші параметри. Дотримуйтеся найкращих практик безпеки сервера.

Постійна моніторинг безпеки: Встановіть систему постійного моніторингу безпеки, яка виявлятиме небезпеку та попереджатиме про можливі проблеми. Приймайте вчасні заходи для виправлення виявлених проблем.

Не забувайте, що безпека є постійним процесом. Важливо перевіряти безпеку вашого сайту регулярно та приймати відповідні заходи для її поліпшення.

2.7.2. СИСТЕМНЕ ЛОГУВАННЯ

Системне логування в Drupal 8 дозволяє відстежувати та аналізувати події, які відбуваються на вашому сайті. Це може бути корисно для виявлення помилок, проблем безпеки, відстеження активності користувачів та іншої важливої інформації. Ось детальний огляд системного логування в Drupal 8:

Конфігурація логування: Ви можете налаштувати параметри системного логування в файлі `settings.php` або за допомогою інтерфейсу адміністративної панелі. У файлі `settings.php` ви можете задати рівень логування, шлях до лог-файлу та інші параметри. В адміністративній панелі ви можете налаштувати рівень логування, вибрати типи подій для логування та збереження лог-файлів.

Рівні логування: Drupal 8 підтримує кілька рівнів логування, які дозволяють контролювати, які події будуть логуватися. Рівні логування включають:

- None: Логування вимкнене.
- Errors and warnings: Логування тільки помилок та попереджень.
- All messages: Логування всіх повідомлень, включаючи інформаційні повідомлення.

Типи подій: Ви можете вибрати, які типи подій будуть логуватися. Наприклад, ви можете логувати помилки, попередження, інформаційні повідомлення, запити до бази даних, події користувачів та багато іншого. Це дозволяє вам налаштувати логування тільки для потрібних подій і уникнути зайвого обсягу лог-файлів.

Лог-файли: Drupal 8 зберігає лог-файли в заданому каталозі на сервері. Ви можете вибрати шлях до каталогу лог-файлів та формат іменування файлів. Це дозволяє вам легко знайти і аналізувати лог-файли.

Аналіз лог-файлів: Після збереження лог-файлів ви можете аналізувати їх для виявлення помилок, проблем безпеки або іншої важливої інформації. Ви можете використовувати сторонні інструменти аналізу логів або розробляти власні скрипти для обробки лог-файлів.

Моніторинг подій: Drupal 8 також надає інтерфейс адміністративної панелі для перегляду останніх подій. Ви можете переглянути логи подій, включаючи помилки, попередження та інші типи подій, що були логовані. Це дає вам можливість відстежувати активність на вашому сайті безпосередньо через інтерфейс Drupal.

Розширення логування: У Drupal 8 є також можливість розширити систему логування за допомогою сторонніх модулів. Ці модулі можуть додавати нові типи подій, фільтри логування, налаштування лог-файлів та інші функції.

Системне логування в Drupal 8 дозволяє вам отримати цінну інформацію про стан вашого сайту, виявити проблеми безпеки та допомогти вам відновити роботу сайту у випадку виникнення проблем. Важливо налагоджувати належні налаштування логування та регулярно перевіряти лог-файли для забезпечення безпеки та стабільності вашого сайту.

Для доступу до системного логу Drupal 8 за допомогою Drush використовується команда `drush watchdog-show`. Ця команда дозволяє переглядати записи системного логу безпосередньо з командного рядка. Ось кілька прикладів використання команди `drush watchdog-show`:

Вивести останні записи системного логу:

drush watchdog-show

Вивести останні 10 записів системного логу:

drush watchdog-show --count=10

Вивести записи системного логу за певний часовий період:

drush watchdog-show --from="2019-01-01 00:00:00" --to="2019-12-31 23:59:59"

Вивести записи системного логу за певним рівнем логування:

drush watchdog-show --severity=error

Фільтрація записів системного логу за певним текстом:

drush watchdog-show --filter="error"

Вивести детальну інформацію про певний запис системного логу за його ID:

drush watchdog-show --id=123

Ви можете використовувати різні комбінації параметрів команди `drush watchdog-show`, щоб відображати, фільтрувати та аналізувати записи системного логу в Drupal 8. Зверніть увагу, що доступ до системного логу через Drush може знадобитися під адміністративними правами.

2.7.3. УПРАВЛІННЯ ПРАВАМИ ДОСТУПУ ДО ФАЙЛОВОЇ СИСТЕМИ

Управління доступом до файлової системи в Drupal 8 дозволяє контролювати, які користувачі мають доступ до файлів на вашому сайті. Це важлива функціональність для забезпечення безпеки та обмеження доступу до конфіденційної інформації. Ось детальний огляд управління доступом до файлової системи в Drupal 8:

Папка файлів: Drupal 8 має папку за замовчуванням для збереження завантажених файлів, яка зазвичай розташовується в папці `sites/default/files`. Ви можете налаштувати іншу папку для збереження файлів, якщо це необхідно.

Права доступу до папки файлів: Важливо налаштувати правильні права доступу до папки файлів на вашому сервері. Папка повинна бути доступною для запису тільки для веб-сервера та обмежена для інших користувачів. Це допоможе запобігти несанкціонованому доступу до файлів.

Типи файлів: Drupal 8 дозволяє налаштовувати типи файлів і обмеження для кожного типу. Ви можете визначити дозволені розширення файлів, максимальний розмір файлу та інші параметри. Це дозволяє контролювати, які типи файлів можуть бути завантажені на ваш сайт.

Поля файлів: У Drupal 8 ви можете використовувати поля файлів у контентних типах для завантаження файлів. Ви можете налаштувати обмеження доступу до полів файлів для різних ролей користувачів. Наприклад, ви можете дозволити лише адміністраторам завантажувати певні типи файлів.

Політика збереження файлів: Drupal 8 надає можливість зберігати завантажені файли відразу в базі даних або на файловій системі. Ви можете вибрати найбільш підходящий спосіб збереження залежно від ваших потреб. Збереження файлів в базі даних може бути зручним для менших проектів, але збереження на файловій системі може бути ефективнішим для великих обсягів файлів.

Управління правами доступу: Drupal 8 надає розширений механізм управління правами доступу до файлів. Ви можете встановлювати права доступу до файлів для різних ролей користувачів і визначати, хто має право переглядати, завантажувати або видаляти файли. Це дозволяє гнучко контролювати доступ до файлової системи.

Модулі безпеки файлів: У Drupal 8 є декілька модулів безпеки файлів, які допомагають забезпечити безпеку ваших завантажених файлів. Наприклад, модуль File Security дозволяє контролювати доступ до файлів засобами `.htaccess`, що забезпечує додатковий рівень безпеки.

Перевірка безпеки: Важливо перевіряти безпеку вашої файлової системи на регулярній основі. Це включає перевірку прав доступу до папки файлів, перевірку прав доступу до бази даних та перевірку наявності оновлень для модулів безпеки файлів.

Загалом, управління доступом до файлової системи в Drupal 8 дозволяє вам забезпечити безпеку вашого сайту та контролювати доступ до завантажених файлів. Важливо налаштовувати належні права доступу, використовувати типи файлів і поля файлів з обмеженнями та перевіряти безпеку файлової системи на регулярній основі.

Приватна файлова система в Drupal 8 є одним з способів контролювати доступ до завантажених файлів на вашому сайті. Вона дозволяє зберігати файли в захищеній області, недоступній для прямого доступу з веб-браузера. Ось детальний огляд приватної файлової системи в Drupal 8:

Налаштування шляху до приватної папки: У Drupal 8 ви можете налаштувати шлях до папки приватної файлової системи. За замовчуванням, приватна папка знаходиться в папці `sites/default/files/private`. Ви можете змінити цей шлях, якщо це необхідно, у файлі `settings.php`.

Заборона прямого доступу: При використанні приватної файлової системи, файли знаходяться в захищеній області, недоступній для прямого доступу з веб-браузера. Це означає, що користувачі не можуть отримати доступ до файлів, вводячи пряму URL-адресу файлу в браузері.

Контроль доступу: Для отримання доступу до файлів приватної файлової системи, користувачам потрібно мати відповідні права доступу. Ви можете контролювати доступ до файлів, використовуючи систему прав доступу Drupal, таку як ролі користувачів і налаштування дозволів.

Завантаження і завантаження файлів: Для завантаження файлів до приватної файлової системи використовуються стандартні функції завантаження файлів Drupal. При завантаженні файлів до приватної папки, вони автоматично зберігаються в безпечній області і отримують обмеження доступу.

Захист від прямих посилань: Оскільки файли в приватній файловій системі недоступні для прямого доступу, ви можете використовувати спеціальні методи для відображення або завантаження цих файлів на вашому сайті.

Наприклад, ви можете використовувати поля файлів у контентних типах або програмно створювати лінки для скачування файлів.

Безпека і конфіденційність: Використання приватної файлової системи дозволяє забезпечити більшу безпеку та конфіденційність для ваших файлів. Важливо налаштувати належні права доступу до приватної папки, щоб запобігти несанкціонованому доступу до файлів.

Приватна файлова система в Drupal 8 є потужним інструментом для збереження і керування завантаженими файлами з контролем доступу і безпекою. Вона дозволяє зберігати конфіденційні файли в безпечній області і контролювати доступ до них за допомогою системи прав доступу Drupal.

2.7.4. ПОТЕНЦІЙНО НЕБЕЗПЕЧНІ ЕЛЕМЕНТИ КОНФІГУРАЦІЙ

У Drupal 8 існує кілька потенційно небезпечних елементів конфігурації, які потребують особливої уваги і заходів безпеки. Ось кілька з них:

Незахищена папка з файлами: Якщо папка з файлами (`sites/default/files`) не має належних прав доступу, це може призвести до несанкціонованого доступу до завантажених файлів, виконання зловмисного коду або втрати конфіденційної інформації. Впевніться, що права доступу до папки з файлами налаштовані належним чином.

Неактуалізовані модулі: Використання застарілих або небезпечних модулів може створювати ризики безпеки для вашого сайту. Важливо регулярно оновлювати модулі до останніх версій та видаляти ті, які вже не використовуються.

Незахищена база даних: Якщо база даних Drupal 8 не має належних заходів безпеки, це може призвести до несанкціонованого доступу до даних, витоку конфіденційної інформації або втрати даних. Впевніться, що встановлені сильні паролі для бази даних і що база даних захищена від несанкціонованого доступу зовні.

Вразливості PHP: Якщо використовується вразлива версія PHP, це може призвести до виконання зловмисного коду або атаки на ваш сайт. Впевніться, що встановлена остання стабільна версія PHP і регулярно оновлюйте її для отримання оновлень безпеки.

Загрози захоплення сесій: Неправильна настройка сесій може призвести до загрози захоплення сесій і зловживання привілеїв користувача. Переконайтеся, що сесії налаштовані правильно і використовуються безпечні методи для збереження сесій.

XSS атаки: Вразливості, пов'язані з міжсайтовим скриптингом (XSS), можуть дозволити зловмиснику вставляти шкідливий код на сторінках вашого сайту. Використовуйте безпечні фільтри і санітайзери, щоб захистити свій сайт від XSS атак.

SQL-ін'єкції: Неправильна обробка вхідних даних може призвести до SQL-ін'єкцій, які дозволяють зловмисникам отримати доступ до бази даних і виконувати шкідливі операції. Використовуйте параметризовані запити або екрануйте вхідні дані, щоб захистити свою базу даних від SQL-ін'єкцій.

Важливо бути постійно в курсі потенційних небезпек і оновлювати систему, модулі та заходи безпеки відповідно. Регулярні аудити безпеки, встановлення захисних заходів і відслідковування оновлень допоможуть забезпечити безпеку вашого сайту на Drupal 8.

2.7.5. ОНОВЛЕННЯ ТА ПАТЧІ

Оновлення Drupal 8 - це процес оновлення вашого сайту на останню версію Drupal 8.x.x. Це важливий крок для забезпечення безпеки, функціональності і виправлення помилок. Ось детальний огляд процесу оновлення Drupal 8:

Резервне копіювання: Перш ніж розпочати процес оновлення, важливо створити повне резервне копіювання вашого сайту, включаючи базу даних і файли. Це забезпечить можливість відновлення в разі проблем або помилок під час оновлення.

Перевірка вимог: Перед оновленням переконайтеся, що ваш сервер відповідає вимогам нової версії Drupal 8. Перевірте, чи встановлені потрібні версії PHP, MySQL або інших необхідних залежностей.

Підготовка до оновлення: Перед оновленням виконайте кілька підготовчих кроків. Зробіть оновлення всіх встановлених модулів і тем до останніх версій сумісних з новою версією Drupal 8. Переконайтеся, що ваш темплейт і будь-які власні модифікації коду сумісні з новою версією.

Оновлення ядра: Оновлення ядра Drupal 8 включає заміну файлів ядра новими версіями. Існує кілька способів оновлення ядра, включаючи використання інтерфейсу адміністрування, командного рядка або інструментів керування кодом, таких як Git.

Оновлення бази даних: Після оновлення ядра Drupal 8 вам може знадобитися оновлення бази даних. Drupal автоматично виявляє потрібні зміни в схемі бази даних і пропонує виконати оновлення. Важливо уважно прочитати інструкції та зробити резервну копію бази даних перед оновленням.

Перевірка функціональності і виправлення проблем: Після оновлення перевірте роботу вашого сайту, включаючи функціональність, візуальний вигляд, модулі і теми. Перевірте, чи працюють всі функції правильно, чи не виникає помилок або конфліктів.

Оновлення модулів і тем: Після оновлення ядра Drupal 8 переконайтеся, що всі встановлені модулі і теми також оновлені до сумісних версій з новою версією Drupal 8.

Повторення процесу на тестовому сервері: Якщо у вас є тестовий сервер, рекомендується повторити процес оновлення на ньому перед застосуванням на продакшен сервері. Це допоможе виявити та виправити проблеми, які можуть виникнути під час оновлення.

Документація і підтримка: Важливо ознайомитися з документацією та офіційними рекомендаціями Drupal 8 щодо процесу оновлення. Якщо виникають проблеми або запитання, ви можете звернутися до форумів спільноти Drupal або звернутися до команди підтримки.

Важливо виконувати оновлення Drupal 8 регулярно, оскільки це дозволяє отримати оновлення безпеки, нові функції і виправлення помилок. Пам'ятайте, що перед оновленням завжди рекомендується резервне копіювання вашого сайту для безпеки.

Оновлення Drupal 8 за допомогою Composer - це рекомендований спосіб оновлення для проектів, що використовують Composer для керування залежностями. Ось кроки, які потрібно виконати для оновлення Drupal 8 за допомогою Composer:

Зробіть резервну копію: Перш ніж розпочати процес оновлення, важливо створити повне резервне копіювання вашого сайту, включаючи базу даних і файли. Це забезпечить можливість відновлення в разі проблем або помилок під час оновлення.

Оновіть composer.json: Відкрийте файл `composer.json` вашого проекту і змініть версію Drupal на бажану, наприклад `"drupal/core": "^8.9"`. Збережіть зміни.

Виконайте composer update: В командному рядку перейдіть до кореневої папки вашого проекту і виконайте команду `composer update`. Composer перевірить залежності і оновить ядро Drupal 8 та інші модулі до вказаної версії.

Перевірте конфлікти і помилки: Після завершення оновлення Composer перевірить конфлікти залежностей і можливі помилки. Якщо виникають конфлікти або помилки, виправте їх, звернувши увагу на повідомлення, що виводяться в командному рядку.

Оновіть базу даних: Після оновлення Drupal 8 потребує оновлення бази даних для виконання необхідних змін схеми. Виконайте команду `drush updatedb` або

відкрийте веб-інтерфейс сайту, який автоматично виявить потребу в оновленні бази даних.

Перевірте роботу сайту: Після оновлення перевірте роботу вашого сайту, включаючи функціональність, візуальний вигляд, модулі і теми. Переконайтеся, що всі функції працюють правильно і що не виникає помилок або конфліктів

Оновлення модулів і тем: Перевірте, чи всі встановлені модулі і теми сумісні з новою версією Drupal 8. Оновіть їх до останніх сумісних версій.

Важливо регулярно оновлювати Drupal 8, використовуючи Composer, для забезпечення безпеки, отримання нових функцій і виправлення помилок. Уважно слідкуйте за повідомленнями під час оновлення і вчасно виправляйте будь-які помилки або конфлікти, які можуть виникнути під час процесу оновлення.

Використання патчів в Drupal 8 - це процес внесення змін до коду Drupal за допомогою спеціальних файлів патчів. Патчі використовуються для виправлення помилок, внесення змін до функціональності або виконання специфічних змін у коді Drupal. Ось кроки, які потрібно виконати для використання патчів в Drupal 8:

Завантажте патч: Знайдіть необхідний патч для вашої версії Drupal 8. Патчі можна знайти на офіційному сайті Drupal або на спеціальних форумах і репозиторіях.

Перевірте сумісність: Переконайтеся, що патч сумісний з вашою версією Drupal 8. Деякі патчі можуть бути специфічними для певних версій, тому важливо перевірити, чи патч призначений для вашої версії Drupal.

Застосуйте патч: Використовуйте командний рядок або інструмент керування кодом, такий як Git, для застосування патча до вашого проекту Drupal 8. Наприклад, використовуючи команду `git apply`, ви можете застосувати патч до відповідного файлу або директорії.

Перевірте роботу: Після застосування патча перевірте роботу вашого сайту, щоб переконатися, що зміни були успішно внесені і не виникають помилок або конфліктів.

Важливо мати на увазі, що використання патчів вимагає деякого досвіду в роботі з кодом Drupal і знання процесу застосування патчів. Перед застосуванням патча рекомендується завжди робити резервну копію вашого проекту для можливості відновлення в разі проблем. Крім того, патчі можуть стати неактуальними при оновленні Drupal, тому важливо слідкувати за новими версіями і оновленнями, що можуть включати виправлення або зміни, які раніше вносилися за допомогою патчів.

2.7.6. ТИПОВІ МЕТОДИ ВИЯВЛЕННЯ ТА НЕЙТРАЛІЗАЦІЇ ВРАЗЛИВОСТЕЙ

Виявлення та нейтралізація вразливостей в Drupal 8 є важливою частиною забезпечення безпеки вашого сайту. Ось декілька типових методів, які можна використовувати для цих цілей:

Постійне оновлення: Важливо оновлювати Drupal 8 до останньої стабільної версії, а також оновлювати всі встановлені модулі і теми. Розробники Drupal постійно виправляють вразливості та випускають оновлення, щоб забезпечити безпеку сайту. Встановлюйте оновлення якнайшвидше, щоб уникнути можливих атак.

Моніторинг безпеки: Слід слідкувати за повідомленнями про вразливості Drupal 8, офіційними оголошеннями та звітами про безпеку. Використовуйте ресурси, такі як офіційний сайт Drupal, безпечні мережі спільноти та офіційні розсилки, щоб отримувати оновлену інформацію про вразливості та заплановані оновлення.

Використання модулів безпеки: Встановлюйте і налаштовуйте модулі безпеки, які додатково захищають ваш сайт. Наприклад, модуль Security Kit дозволяє заборонити виконання небезпечних файлів, а модуль Password Policy дозволяє задати вимоги до паролів користувачів.

Аудит безпеки: Регулярно проводьте аудит безпеки вашого сайту Drupal 8, виявляючи можливі вразливості та ризики. Це можна зробити самостійно або звернувшись до спеціалізованих фахівців з безпеки. Аудит допоможе виявити слабкі місця та недоліки в конфігурації та розробці вашого сайту.

Контроль доступу: Налаштуйте відповідний контроль доступу для користувачів і адміністраторів вашогосайту. Забезпечте, що кожен користувач має відповідні права доступу, і обмежте права адміністратора лише необхідним функціям і можливостям.

Захист від вразливостей вводу даних: Перевірте вхідні дані, які надсилаються на ваш сайт, щоб уникнути атак типу "Cross-Site Scripting" (XSS) та "SQL Injection". Використовуйте фільтри, валідацію і безпечні методи роботи з базою даних для запобігання таким вразливостям.

Захист від зламів паролів: Вимагайте від користувачів складних паролів, використовуйте методи шифрування паролів та встановлюйте обмеження на спроби невдалих авторизацій.

Для аналізу вразливостей в Drupal 8 і забезпечення безпеки сайту ви можете використовувати наступні модулі:

Security Review: Цей модуль аналізує ваш сайт на предмет відповідності до безпекових стандартів Drupal і надає звіт з порадами щодо покращення безпеки. Він перевіряє такі аспекти, як налаштування конфігурації, додаткові модулі, ролі користувачів, потрібні оновлення та інші важливі фактори.

Drupalgeddon: Це набір модулів та інструментів, які допомагають виявляти потенційні вразливості, зокрема ті, які пов'язані з вразливістю Drupalgeddon. Цей набір містить модуль Drupalgeddon Detection, який перевіряє ваш сайт на предмет вразливостей і надає рекомендації щодо захисту.

Paranoia: Цей модуль допомагає виявити можливі вразливості безпеки в вашому сайті. Він перевіряє різні аспекти, такі як налаштування безпеки, звички розробки і використання модулів, і надає рекомендації щодо покращення безпеки.

Hacked!: Цей модуль допомагає виявити можливі вторгнення або зміни в коді вашого сайту, що можуть бути наслідком вразливостей безпеки. Він порівнює вашу установлену версію Drupal з офіційними випусками і знаходить потенційно змінений або пошкоджений код.

Security Kit: Цей модуль дозволяє налаштовувати додаткові безпекові політики для вашого сайту, такі як заборона виконання певних файлів або обмеження доступу до певних типів файлів. Він також надає можливість встановити HTTP-заголовки безпеки для запобігання певним атакам.

Це лише кілька прикладів модулів, які можна використовувати для аналізу вразливостей в Drupal 8. Варто також зазначити, що забезпечення безпеки - це постійний процес, і регулярне оновлення вашого сайту та виконання найкращих практик безпеки є ключовими для запобігання вразливостям.

Питання для самоконтролю:

1. Як проводити аудит безпеки?
2. Як використовувати системний лог?
3. Для чого використовуються патчі?
4. Як встановлювати оновлення?
5. Які налаштування можуть бути потенційними причинами небезпеки?

3. ОСНОВИ РОЗРОБКИ В CMF DRUPAL (MODULE AND THEME DEVELOPMENT)

Drupal 8 - це потужний веб-фреймворк, який базується на PHP і має модульну архітектуру. Розглянемо основні моменти:

Об'єктно-орієнтованість: Drupal 8 використовує об'єктно-орієнтований підхід до програмування. Багато компонентів, класів та функцій Drupal 8 реалізовані як об'єкти, що дозволяє більшу гнучкість та повторне використання коду.

Хороша організація коду: Код Drupal 8 добре організований і має структуру, що спрощує розробку та зрозумілість коду. Кожен функціональний аспект Drupal 8 реалізований як модуль, що дозволяє легко розширювати функціональність та підтримувати код.

Symfony компоненти: Drupal 8 використовує Symfony, один з найпопулярніших PHP-фреймворків, як базовий компонент для реалізації багатьох основних функцій. Це забезпечує більшу стабільність, широкий вибір готових компонентів та допомагає підтримувати сучасні стандарти розробки.

Шаблонування з використанням Twig: Drupal 8 використовує Twig - потужний і сучасний двіжок шаблонізації, що дозволяє легко розділити логіку від представлення. Twig забезпечує зрозумілу та зручну синтаксичну структуру для шаблонів.

Модульна архітектура: Drupal 8 побудований на основі модульної архітектури, що дозволяє додавати нові функціональність шляхом створення та використання модулів. Кожен модуль може мати свою власну структуру, функціонал та розширювати базовий функціонал Drupal 8.

API для розширення: Drupal 8 надає багатий набір API для розширення функціональності. Це включає хуки, події, сервіси, роутинг, форми та інші механізми, що дозволяють розробникам розширювати та змінювати функціональність фреймворка.

Безпека: Drupal 8 надає широкі можливості для забезпечення безпеки веб-сайту. Це включає вбудовані механізми захисту від атак, обробку вхідних даних, захист доступу до контенту та інші безпекові функції.

Документація та спільнота: Drupal 8 має велику та активну спільноту розробників, яка надає документацію, підтримку та взаємодію з іншими розробниками. Це допомагає розробникам зрозуміти фреймворк, розв'язувати проблеми та швидше розвиватись у сфері Drupal 8.

Це лише кілька аспектів Drupal 8 як фреймворка з точки зору коду. Drupal 8 надає розробникам потужні інструменти та можливості для створення складних та потужних веб-сайтів і додатків.

3.1. ПОНЯТТЯ ПРО МОДУЛІ ТА ТЕМИ. СТАНДАРТИ КОДУВАННЯ. ВІДЛАГОДЖЕННЯ

В Drupal 8, модуль - це основна одиниця функціональності, яка додається до системи для розширення її можливостей. Модулі дозволяють додавати нові функції, змінювати існуючі функції, розширювати можливості адміністрування та забезпечувати інтеграцію з іншими системами.

Поняття модуля в Drupal 8:

Структура модуля: Кожен модуль має свою структуру каталогів та файлів. Типова структура модуля включає файли, такі як .info.yml (який містить інформацію про модуль, його залежності, версію тощо), .module (який містить PHP-код модуля), .routing.yml (який визначає шляхи модуля) та .theme (який містить тематичні файли, якщо модуль працює зі стилізацією).

Хуки: Модулі можуть визначати хуки, які дозволяють виконати дії у певні моменти життєвого циклу Drupal. Хуки дозволяють іншим модулям реагувати на події, що відбуваються в системі або змінювати її поведінку. Наприклад, модуль може визначити хук, який виконується при створенні нового контенту, і виконати певні додаткові дії.

Роутинг: Drupal 8 використовує систему роутингу для маршрутизації запитів. Кожен модуль може визначити свої власні шляхи (routes), які вказують, як система повинна обробляти певні URL-адреси. Це дозволяє модулям визначати свої власні сторінки, дії та обробники.

Сервіси: Drupal 8 використовує систему сервісів для управління залежностями та спільним використанням функціональності між модулями. Модуль може визначити свої власні сервіси, які можуть бути використані іншими модулями для отримання функціональності, яку вони надають.

Тематика: Модуль може включати тематичні файли (шаблони), які використовуються для відображення вмісту на сторінках. Це дозволяє модулю змінювати вигляд та стилізацію контенту.

Налаштування: Модуль може визначати свої власні налаштування, які можуть бути змінені адміністратором сайту. Це дозволяє користувачам настроювати функціональність модуля під свої потреби.

Залежності: Модуль може мати залежності від інших модулів або бібліотек. Це дозволяє модулю використовувати функціональність інших модулів або розширення зовнішніх бібліотек.

Модулі в Drupal 8 дозволяють розробникам додавати нові функції та розширювати можливості системи. Вони надають структурований підхід до розробки та дозволяють розширювати та змінювати Drupal згідно з вимогами проекту.

При створенні кастомних модулів в Drupal 8 існує декілька правил і бест практик, яких варто дотримуватись. Ось деякі з них:

Неймінг: Назва модуля повинна бути унікальною і відповідати певним правилам. Зазвичай, назва модуля складається з маленьких латинських літер і символу підкреслення, наприклад, "my_module".

Файл .info.yml: Кожен модуль повинен мати файл модуля .info.yml, в якому вказується основна інформація про модуль, така як назва, версія, залежності та інші налаштування.

Об'єктно-орієнтований підхід: Drupal 8 рекомендує використовувати об'єктно-орієнтований підхід до розробки модулів. Це означає використання класів і об'єктів для реалізації функціональності модуля.

Файл .module: Файл .module є основним файлом модуля, в якому реалізовується функціональність. У цьому файлі можна визначати хуки, реалізовувати функції, налаштувати роутинг та інші операції.

Структура каталогів: Код модуля повинен бути організований у відповідну структуру каталогів. Рекомендована структура включає каталоги, такі як "src" для класів, "Controller" для контролерів, "Form" для форм, "Plugin" для плагінів та інші відповідно до функціональних блоків модуля.

Кодування: Дотримання стандартів кодування, таких як PSR-4, допоможе зробити ваш код зрозумілим та підтримуваним. Рекомендується використовувати стандартні практики та змінні імена зрозумілі для розробників.

Модульність: Рекомендується створювати кожен модуль з мінімальною функціональністю та уникати надмірної залежності між модулями. Це дозволяє зберігати код модуля більш зрозумілим, підтримуваним та переносимим.

Документація: Важливо додавати коментарі та документацію до вашого коду, щоб розробники, які будуть працювати з вашим модулем, могли швидко зрозуміти його функціональність та використання.

Тестування: Рекомендується включати тестування в ваш процес розробки. Використання автоматичних тестів допомагає виявляти та усувати помилки та забезпечує більшу стабільність та якість вашого модуля.

Це лише кілька правил і бест практик для створення кастомних модулів в Drupal 8. Кожен проект може мати свої особливості, але дотримання цих загальних рекомендацій допоможе зробити ваш код більш читабельним, підтримуваним та сумісним з екосистемою Drupal.

Drupal 8 використовує стандарти кодування, щоб забезпечити єдність, читабельність та підтримку між різними розробниками. Основний стандарт кодування для Drupal 8 - це "Drupal Coding Standards" (стандарти кодування Drupal). Ось деякі важливі аспекти цих стандартів:

PSR-4: Drupal 8 використовує стандарт PSR-4 для автозавантаження класів. Це означає, що класи повинні бути організовані у відповідну структуру каталогів, а імена класів повинні відповідати правилам пространства імен PSR-4.

Кодування: Стандарти кодування Drupal 8 включають в себе використання відступів з 4 пробілами, використання високого регістру для функцій, використання нижнього підкреслення для змінних та методів, використання дужок умовних виразів, відсутність табуляції тощо.

Довжина рядків: Максимальна довжина рядків в коді Drupal 8 обмежена 80 символами. Це сприяє збереженню читабельності коду та зменшенню потреби у горизонтальному прокручуванні.

Коментарі: Код в Drupal 8 повинен бути добре задокументований за допомогою коментарів. Коментарі допомагають розробникам зрозуміти функціональність, ціль та використання коду.

Правила іменування: У Drupal 8 існують певні правила іменування, які дотримуються при назвах змінних, функцій, класів та інших елементів коду. Наприклад, змінні повинні бути названі в camelCase, класи - в PascalCase, а функції - в snake_case.

Дотримання стандартів кодування Drupal 8 рекомендується для забезпечення читабельності, підтримки та співпраці з іншими розробниками. У Drupal 8 також існує інструментарій для перевірки відповідності коду стандартам, такий як "Coder" та "PHP

Code Sniffer". Використання цих інструментів може спростити процес перевірки та виправлення відхилень від стандартів кодування Drupal 8.

При створенні модулів для Drupal 8 рекомендується використовувати лінтери та сніфери для перевірки відповідності вашого коду стандартам кодування та виявлення потенційних проблем. Ось декілька популярних лінтерів та сніферів, які можна використовувати:

PHP Code Sniffer (PHPCS): PHP Code Sniffer є потужним інструментом для перевірки стандартів кодування. Він перевіряє ваш PHP-код на відповідність стандартам кодування, таким як PSR-1, PSR-2 та Drupal Coding Standards. PHPCS зазвичай використовується з набором правил, таким як "Drupal Coder", що дозволяє перевіряти відповідність коду стандартам Drupal.

PHPStan: PHPStan є статичним аналізатором PHP-коду, який виявляє потенційні проблеми, такі як помилки типізації, неправильне використання функцій, неправильне використання змінних та інше. Він дозволяє забезпечити більшу стабільність та якість вашого коду.

ESLint: Якщо ви використовуєте JavaScript у своєму модулі, ESLint є потужним інструментом для перевірки вашого JavaScript-коду на відповідність стандартам кодування та виявлення потенційних проблем.

CSS Lint: Якщо ви використовуєте CSS у своєму модулі, CSS Lint допоможе перевірити ваш CSS-код на відповідність стандартам кодування та рекомендаціям для кращої практики.

Використання лінтерів та сніферів під час розробки модулів для Drupal 8 допоможе забезпечити високу якість коду, виконання стандартів кодування та покращення читабельності та підтримуваності вашого коду.

Тема в Drupal 8 - це набір файлів, що визначають зовнішній вигляд вашого сайту. Вона визначає, як будуть відображатися сторінки, блоки, вміст та інші елементи на вашому сайті.

Тема в Drupal 8 складається з різних компонентів:

Шаблони (Templates): Шаблони визначають структуру та розмітку HTML-сторінок. Існують різні шаблони для різних типів сторінок, таких як головна сторінка, сторінка вмісту, блоки, форми тощо. Шаблони мають розширення `.html.twig` і використовують спеціальну синтаксичну розмітку Twig для виведення динамічних даних.

Стилі (Stylesheets): Стилі визначають зовнішній вигляд сторінок та елементів на сайті. Вони можуть бути написані з використанням CSS-розмітки або препроцесорів CSS, таких як SASS або LESS. Тема може включати один або кілька CSS-файлів для стилізації різних елементів.

JavaScript-файли: JavaScript-файли використовуються для додавання інтерактивності та функціональності до вашого сайту. Вони можуть містити власний JavaScript-код або підключати зовнішні бібліотеки.

Конфігураційні файли: Тема може містити конфігураційні файли, які визначають деякі параметри теми, такі як назва, версія, автор тощо. Один з основних конфігураційних файлів - `mytheme.info.yml`, в якому вказується основна інформація про тему.

Теми можуть бути розроблені з нуля або ви можете використовувати готові теми, які доступні у Drupal 8. Ви можете модифікувати готову тему, виконавши зміни в шаблонах, стилях та JavaScript-файлах, щоб відповідати вимогам вашого проекту.

Теми в Drupal 8 дозволяють вам змінювати зовнішній вигляд вашого сайту без втручання у базовий функціонал Drupal. Ви можете створювати унікальний дизайн, відповідний вашим потребам та бренду, зберігаючи при цьому всю функціональність та можливості, які надає Drupal 8.

Створення тем в Drupal 8 дозволяє налаштовувати зовнішній вигляд вашого сайту. Ось детальна інструкція щодо створення теми в Drupal 8:

Створіть каталог для вашої теми: Спочатку потрібно створити каталог для вашої теми у каталозі `themes` вашого Drupal-сайту. Назвіть його відповідно до назви вашої теми, наприклад, `mytheme`.

2. Створіть файли теми: У каталозі вашої теми створіть основні файли:

- `mytheme.info.yml`: Це YAML-файл, що містить основну інформацію про вашу тему, таку як назва, версія, автор і т. д. Додайте відповідні ключі та значення до цього файлу.
- `mytheme.libraries.yml`: Цей файл використовується для оголошення бібліотек, які використовуються вашою темою, такі як CSS- та JavaScript-файли.
- `mytheme.theme`: Це PHP-файл, який використовується для оголошення функцій-хуків та розширення функціональності вашої теми.

Оголоште базові файли теми: Ваша тема повинна мати основні файли, такі як:

- `html.twig`: Цей шаблон використовується для огортання всього контенту вашого сайту. Ви можете визначити розмітку HTML та підключати різні регіони та шаблони.
- `page.twig`: Цей шаблон використовується для відображення сторінок вашого сайту. Ви можете розміщувати контент та регіони на сторінці.
- `node.twig`: Цей шаблон використовується для відображення вмісту типу "вузол" (node). Ви можете визначити, як вміст вузла буде відображатися на сторінці.

Додайте CSS та JavaScript: У файлі `mytheme.libraries.yml` визначте ваші бібліотеки CSS та JavaScript. Ви можете використовувати зовнішні бібліотеки або створювати власні стилі та скрипти.

Налаштуйте розміщення блоків: Ви можете налаштувати розміщення блоків в вашій темі, використовуючи файл `mytheme.theme` та функцію `mytheme_preprocess_region`. Це дозволяє вам впливати на розміщення блоків у ваших регіонах.

Налаштуйте зображення та стилі: Ви можете налаштувати зображення та стилі вашої теми, використовуючи CSS та файл `mytheme.theme`. Наприклад, ви можете визначити стилі для заголовків, фонові зображення та інші елементи.

Активуйте тему: У файлі `mytheme.info.yml` встановіть ключ `base theme` на `stable` та активуйте вашу тему, перейшовши до панелі адміністрування Drupal та вибравши вашу тему в розділі "Адміністрування тем".

Це загальна структура та процес створення теми в Drupal 8. Ви можете детальніше налаштовувати та розширювати вашу тему, використовуючи додаткові файли та функціонал Drupal 8.

Twig - це мова шаблонів, яка використовується в Drupal 8 для відображення динамічних даних у шаблонах. Вона є частиною Symfony, потужного PHP-фреймворка, який використовується в Drupal.

Основні особливості та можливості Twig в Drupal 8:

Синтаксис: Twig має чистий та простий синтаксис, який легко читати та розуміти. Він використовує функції, фільтри, блоки та змінні для маніпулювання та відображення даних.

Розділення логіки та представлення: Twig дотримується принципу "розділення логіки та представлення". Це означає, що ви можете відокремити логіку вашого сайту в

контролери, а шаблони Twig використовувати для відображення даних без прямого виконання коду PHP в шаблонах.

Вбудовані функції та фільтри: Twig надає багато вбудованих функцій та фільтрів для роботи з даними у шаблонах. Наприклад, ви можете використовувати фільтр `date` для форматування дати, функцію `trans` для перекладу рядків або фільтр `length` для отримання довжини масиву.

Керування структурою даних: Twig дозволяє легко керувати структурою даних у шаблонах. Ви можете використовувати умовні конструкції, цикли та ітерації для обробки та відображення даних згідно потреб вашого шаблону.

Наслідування та блоки: Twig підтримує механізм наслідування, який дозволяє створювати базовий шаблон, який можна розширювати та наслідувати в інших шаблонах. Ви також можете використовувати блоки, щоб перезаписувати або доповнювати контент у вищих рівнях шаблону.

Twig використовується в Drupal 8 для відображення шаблонів сторінок, блоків, форм, списків тощо. Ви можете створювати власні шаблони та використовувати функціонал Twig для зручного відображення та маніпулювання даними у ваших Drupal 8 проектах.

Питання для самоконтролю:

1. Як організована кодова база в Drupal 8?
2. Що таке модуль?
3. Що таке код стандарти і як їх використовувати?
4. Що таке лінтери та сніфери, як їх використовувати?
5. Що таке тема?
6. Що таке Twig та для чого він призначений?

3.2. РОЗРОБКА МОДУЛІВ

Процес створення модуля в Drupal 8 може бути розділений на кілька основних кроків. Визначимо ці кроки:

Планування: Перш ніж розпочати створення модуля, рекомендується зробити план роботи. Визначте мету вашого модуля, його функціональні можливості та необхідні компоненти.

Створення папки модуля: У кореневій директорії вашого Drupal сайту створіть папку з назвою вашого модуля, наприклад, "my_module".

Створення info.yml файлу: У папці вашого модуля створіть файл з розширенням `.info.yml`. Цей файл визначає основну інформацію про ваш модуль, таку як назва, версія, опис тощо.

Створення модульного файлу: У тій же папці створіть модульний файл з розширенням `.module`. Цей файл виконується Drupal під час завантаження модуля і містить логіку вашого модуля, таку як обробники подій, функції ініціалізації та інше.

Реєстрація модуля: Відкрийте файл `my_module.info.yml` і додайте необхідну інформацію про ваш модуль, наприклад, назву, версію, залежності, шлях до модульного файлу тощо.

Реалізація функціоналу: Додавайте необхідний функціонал у модульний файл, використовуючи Drupal API та різні хуки, які дозволяють взаємодіяти з ядром та іншими модулями.

Реєстрація маршрутів: У папці модуля створіть файл `my_module.routing.yml`. В цьому файлі ви можете визначити маршрути вашого модуля та пов'язані з ними контролери або дії.

Створення шаблонів: У папці модуля створіть підпапку `templates` та в ній створіть шаблони `.html.twig`, які будуть використовуватись для відображення даних вашого модуля.

Тестування: Рекомендується створити тести для вашого модуля, щоб переконатися, що функціонал працює правильно та відповідає очікуванням.

Документація: Не забудьте документувати ваш модуль, описати його функціонал, використовані технології та інше. Це допоможе іншим розробникам або вам самим при подальшій роботі з модулем.

Розгортання модуля: Завершивши розробку, ви можете розгорнути свій модуль на вашому Drupal сайті шляхом встановлення та активації його через адміністративний інтерфейс Drupal.

Це загальний опис процесу створення модуля в Drupal 8. Зважайте, що кожен проект може мати свої особливості, і варто дотримуватись кращих практик та стандартів розробки Drupal при створенні власного модуля.

3.2.1 СТРУКТУРА ФАЙЛІВ ТА ДИРЕКТОРІЙ МОДУЛЯ

Структура папок модуля в Drupal 8 допомагає організувати ваш код та ресурси таким чином, щоб вони були доступні для Drupal і легко керовані. Ось загальна структура папок модуля в Drupal 8:

Папка модуля: Створіть папку з назвою вашого модуля в кореневій папці "modules" вашого Drupal сайту. Наприклад, `modules/custom/my_module`.

Файл .info.yml: В кореневій папці вашого модуля створіть файл з розширенням `.info.yml`. Цей файл містить основну інформацію про ваш модуль, таку як назву, опис, версію, автора, залежності та інші метадані.

Папка src: В папці модуля створіть підпапку з назвою "src". Ця папка містить PHP-код вашого модуля, такі як контролери, сервіси, форми та інші класи.

Папка templates: В папці модуля створіть підпапку з назвою "templates". Вона використовується для зберігання шаблонів Twig, які використовуються для відображення даних вашого модуля.

Папка config: В папці модуля створіть підпапку з назвою "config". Вона використовується для зберігання конфігураційних файлів вашого модуля, таких як файл `.yml` зі значеннями за замовчуванням.

Папка tests: В папці модуля створіть підпапку з назвою "tests". Вона використовується для зберігання тестів вашого модуля для автоматизованого тестування.

Файл .module: В кореневій папці вашого модуля створіть файл з розширенням `.module`. Цей файл виконується Drupal при завантаженні модуля і містить логіку вашого модуля, таку як обробники подій, функції ініціалізації та інше.

Інші ресурси: Залежно від потреб вашого модуля, ви також можете створити додаткові папки для зберігання ресурсів, таких як стилі CSS, скрипти JavaScript, зображення та інші.

Це загальна структура папок модуля в Drupal 8. Зважайте, що ви можете мати додаткові папки або організувати ваш код по-різному, в залежності від потреб вашого проекту. Проте, рекомендується дотримуватись кращих практик та стандартів Drupal для організації вашого модуля.

Для організації обробки шляхів в Drupal 8 використовується PSR-4 (PHP Standard Recommendation 4), який є стандартом автозавантаження класів у PHP, і дозволяє організувати код проекту у логічну структуру папок та простори імен. PSR-4 визначає правила, за якими PHP-класи автоматично завантажуються на основі їх імені та місця розташування у файловій системі.

В Drupal 8 використовується PSR-4 для автозавантаження модулів, тем та інших компонентів. Це дозволяє зручно організувати код вашого модуля або теми за допомогою папок і просторів імен, що спрощує управління проектом та підтримку автозавантаження класів.

Основні принципи PSR-4:

- Простір імен повинен відображатись на структуру папок: Назви просторів імен відображаються у структурі папок. Наприклад, простір імен `MyModule\SubNamespace` відповідатиме папці `modules/custom/my_module/src/SubNamespace`.
- Кожен клас повинен мати свій файл: Кожен клас повинен бути розташований у відповідному файлі з таким же іменем, як у класу. Наприклад, клас `MyModule\SubNamespace\MyClass` буде розміщений у файлі `MyClass.php`.
- Головний простір імен відповідає кореневій папці: Головний простір імен відображається у кореневій папці. Наприклад, простір імен `MyModule` відповідає папці `modules/custom/my_module/src`.

Для використання PSR-4 у Drupal 8, ви повинні визначити відповідні налаштування у файлі `composer.json` вашого модуля або теми. Наприклад:

```
{
  "autoload": {
    "psr-4": {
      "MyModule\\": "src/"
    }
  }
}
```

Після оновлення `composer.json` виконайте команду `composer dump-autoload`, щоб згенерувати файли автозавантаження. Тепер ваші класи будуть автоматично завантажуватись згідно правил PSR-4.

Використання PSR-4 дозволяє зручно організовувати код вашого модуля або теми в Drupal 8 і покращує його читабельність та керованість. Крім того, він спрощує процес автозавантаження класів, зменшуючи необхідність вручного підключення файлів.

При створенні назв модулів та файлів в Drupal 8, рекомендується дотримуватись деяких кращих практик. Ось деякі поради:

- Назва модуля: Назва модуля повинна бути унікальною та короткою. Використовуйте зрозумілі та описові назви, які чітко відображають функціональність модуля.
- Простір імен: Використовуйте простір імен для організації класів вашого модуля. Ім'я простору імен повинно відповідати назві модуля і додати додаткові простори для підпапок або функціональних блоків.
- Назва файлу модуля: Назва файлу модуля повинна відповідати назві модуля з малих літер і розширенням `.module`. Наприклад, якщо ваш модуль називається "MyModule", файл модуля повинен мати назву `my_module.module`.
- Назва класу: Назва класу повинна відповідати назві модуля та використовувати CamelCase нотацію. Наприклад, якщо ваш модуль називається "MyModule", клас може мати назву `MyModule`.
- Назви функцій та методів: Використовуйте зрозумілі та описові назви для функцій та методів вашого модуля. Назви повинні бути короткими, але інформативними, щоб легко розібратись у функціональності.
- Кодування файлів: Дотримуйтеся встановленого стилю кодування, такого як PSR-12 або Drupal Coding Standards. Це полегшить спільну роботу над проектом та підтримку коду.
- Організація папок: Дотримуйтеся структури папок, рекомендованої в Drupal 8. Наприклад, PHP-код можна розмістити у папці `src`, шаблони Twig - у папці `templates`, а конфігураційні файли - у папці `config`.
- Коментарі: Додавайте коментарі до свого коду, якщо вони потрібні для пояснення функціональності або складних алгоритмів. Добре оформлені коментарі полегшують зрозуміння коду для вас та інших розробників.

Дотримання цих кращих практик допоможе покращити структуру та зрозумілість вашого коду в Drupal 8. Пам'ятайте, що варто дотримуватись стандартів і рекомендацій Drupal, щоб забезпечити сумісність з іншими модулями та забезпечити легку підтримку вашого коду у майбутньому.

3.2.2. СТРУКТУРА INFO.YML

Файл `info.yml` є одним з ключових файлів для модуля в Drupal 8. Він містить метадані про модуль, такі як назву, версію, залежності та конфігураційні параметри. Ось детальніше про призначення та структуру `info.yml` файлу:

- Призначення `info.yml` файлу:
- Визначення основних метаданих модуля, таких як назва, опис та версія.
- Декларація залежностей модуля від інших модулів або бібліотек.
- Визначення конфігураційних параметрів для модуля.
- Оголошення шляхів до інших файлів модуля, таких як шаблони, ресурси CSS та JavaScript.

Структура `info.yml` файлу:

- `name`: Назва модуля. Наприклад, `name: MyModule`.
- `type`: Тип модуля, який може бути `module`, `theme` або `profile`.
- `description`: Опис модуля.
- `package`: Група або пакет, до якого належить модуль.
- `core_version_requirement`: Версія Drupal, з якою сумісний модуль.
- `dependencies`: Залежності модуля від інших модулів або бібліотек.
- `configure`: Шлях до конфігураційної сторінки модуля.
- `version`: Версія модуля.
- `core`: Мінімальна та максимальна версії ядра Drupal, з якими сумісний модуль.
- `libraries`: Посилання на зовнішні бібліотеки, які використовує модуль.
- `stylesheets`: Список стилів CSS, які потрібно підключити для модуля.
- `scripts`: Список скриптів JavaScript, які потрібно підключити для модуля.
- `settings`: Конфігураційні параметри для модуля.

Наприклад, ось простий приклад структури `info.yml` файлу:

```
name: MyModule
type: module
description: 'My custom module'
core_version_requirement: ^8 || ^9
package: Custom
version: 1.0
core: 8.x
dependencies:
  - node
configure: my_module.settings_form
```

В цьому прикладі модуль називається "MyModule" і він має залежність від модуля "Node". Також визначена версія, опис, пакет та шлях до конфігураційної сторінки модуля.

Важливо відзначити, що `info.yml` файл повинен розташовуватись у кореневій папці модуля або теми.

Структура `info.yml` файлу дозволяє зберігати метадані про модуль та налаштування, які потрібні для його функціонування. Дотримання правильної структури та правильної написання цих метаданих є важливим кроком при створенні модулів в Drupal 8.

При створенні `info.yml` файлу в Drupal 8 варто звернути увагу на деякі важливі моменти. Ось кілька рекомендацій:

Вірний синтаксис: Використовуйте правильний синтаксис YAML для визначення ключів та значень в `info.yml` файлі. Він повинен бути правильно відформатований, включати відступи та розділювачі, які відповідають специфікації YAML.

Унікальне ім'я: Забезпечте, щоб назва модуля, вказана у ключі `name`, була унікальною. Це дозволить уникнути конфліктів з іншими модулями.

Тип модуля: Визначте правильний тип модуля у ключі `type`. Можливі значення: `module`, `theme` або `profile`, залежно від того, чи це модуль, тема або профіль.

Версія Drupal: Встановіть відповідну версію Drupal у ключі `core_version_requirement`. Це дозволить Drupal зрозуміти, з якими версіями ядра модуль сумісний.

Залежності: Якщо ваш модуль залежить від інших модулів або бібліотек, вкажіть їх у ключі `dependencies`. Це допоможе Drupal керувати залежностями і встановлювати їх, якщо модуль активується.

Версія модуля: Вказівка версії модуля у ключі `version` може бути корисною для ведення сліду за релізами і оновленнями вашого модуля.

Конфігураційні параметри: Якщо ваш модуль має конфігураційні параметри, визначте їх у ключі `configuration`.

Шлях до конфігураційної сторінки: Якщо ваш модуль має конфігураційну сторінку, вкажіть її шлях у ключі `configure`. Це дозволить користувачам легко знайти і налаштувати ваш модуль.

Відповідність стандартам: Дотримуйтесь стандартів і найкращих практик Drupal при створенні `info.yml` файлу. Це включає використання відступів, правильне форматування ключів та значень, чітку структуру файлу тощо.

Дотримання цих рекомендацій допоможе забезпечити правильну роботу вашого модуля в Drupal 8 та полегшить його управління і розширення.

3.2.3. РОУТИ

В Drupal 8, роутінг відповідає за маршрутизацію запитів і визначення, який код повинен виконуватись для певних URL. Роутінг є важливою складовою для створення коректних URL-адрес та навігації в Drupal 8. Ось детальніше про роути в Drupal 8:

*.routing.yml` файл:

Роути в Drupal 8 визначаються в `*.routing.yml` файлах, де `*` - це ім'я модуля або теми. Наприклад, `mymodule.routing.yml`.

Визначення роутів:

В `*.routing.yml` файлі визначаються роути за допомогою наступної структури:

```
route_name:
  path: '/url-path'
  defaults:
    _controller: '\Namespace\Controller::method'
    _title: 'Page Title'
  requirements:
    _permission: 'access content'
```

- `route_name`: Унікальне ім'я роуту.
- `path`: URL-шлях для роуту.
- `defaults`: Масив параметрів, які передаються до контролера, такі як `_controller` (клас та метод контролера) та `_title` (заголовок сторінки).
- `requirements`: Вимоги до роуту, такі як `_permission` (необов'язкова перевірка дозволу) або `_role` (вимога до ролі користувача).

Контролери:

Контролери виконують дії, пов'язані з роутом. Вони можуть бути класами PHP, які реалізовують інтерфейс `ControllerInterface`. У роуті контролера виконується метод, визначений у контролері.

Генерація URL:

Для генерації URL на основі роуту можна використовувати функцію `Url::fromRoute()` або хелпер-функцію `Url::fromRoute()` в шаблонах Twig.

Автоматичне отримання URL-шляхів:

В Drupal 8 доступний модуль `pathauto`, який дозволяє автоматично генерувати URL-шляхи для вмісту на основі встановлених шаблонів.

Розширення роутів:

Ви можете розширювати роути і додавати додаткові вимоги або змінювати параметри, використовуючи події `RouteAlterEvent` або хуки.

Переклад роутів:

Роути можна перекладати на різні мови за допомогою модуля `content_translation` або інших модулів для перекладу.

Створення і використання роутів дозволяє вам керувати поведінкою вашого модуля відповідно до визначених URL-шляхів і прав доступу. Це спрощує маршрутизацію запитів і забезпечує гнучкість в роботі з URL-адресами в Drupal 8.

Роути в Drupal 8 можуть бути як статичними так і динаміними. Статичні роути визначаються рядком адреси, який фіксовано задано в routing.yml. Динамічні роути можуть містити як статичну так і динамічну компоненту, зокрема передачу аргументів як частину роута.

У Drupal 8, передача аргументів в роут модуля може бути зручним способом для передачі динамічних даних з URL до контролера або іншого об'єкта, що обробляє роут. Для передачі аргументу в роут модуля, ви можете скористатись наступними кроками:

Визначте роут у `*.routing.yml` файлі вашого модуля з параметром `{argument}`:

```
my_module.my_route:
  path: '/my-path/{argument}'
  defaults:
    _controller: '\Drupal\my_module\Controller\MyController::myMethod'
    _title: 'My Page'
  requirements:
    _permission: 'access content'
```

В контролері або іншому об'єкті, який обробляє роут, додайте аргумент до методу:

```
namespace Drupal\my_module\Controller;
use Symfony\Component\HttpFoundation\Response;

class MyController {

  public function myMethod($argument) {
    // Ваш код обробки залежно від аргументу.
    // Наприклад, можна повернути відповідь залежно від значення аргументу.
    return new Response('Argument value: ' . $argument);
  }
}
```

Тепер, при запиті на `/my-path/value`, `value` буде передано в метод `myMethod` контролера як аргумент. Ви можете використовувати це значення для подальшої обробки або відображення на сторінці.

Пам'ятайте, що при визначенні роутів з аргументами, вам також потрібно вказати тип аргументу у сигнатурі методу контролера (наприклад, `$argument` у прикладі вище).

У Drupal 8, ви можете використовувати права доступу для керування тим, хто має доступ до певних роутів вашого модуля. Це дозволяє вам обмежити доступ до певних функціональних можливостей вашого модуля для певних ролей користувачів або інших умов.

Щоб налаштувати права доступу до роутів, ви можете використовувати ключ `_permission` у файлі `*.routing.yml` вашого модуля. Наприклад:

```
my_module.my_route:
  path: '/my-path'
  defaults:
    _controller: '\Drupal\my_module\Controller\MyController::myMethod'
    _title: 'My Page'
  requirements:
    _permission: 'access content'
```

У цьому прикладі, роут `my_module.my_route` буде доступний лише користувачам, які мають право доступу `access content`. Ви можете замінити `access content` на будь-яке інше право доступу, яке визначено в системі.

Крім `_permission`, ви також можете використовувати інші вимоги (наприклад, `_role`) для керування доступом до роутів. Ви можете комбінувати різні вимоги, щоб досягти необхідного рівня контролю доступу.

Зауважте, що визначення прав доступу до роутів в `*.routing.yml` файлі не єдиний спосіб керування доступом у Drupal 8. Ви також можете використовувати інші методи, такі як рольові фільтри, правила доступу та модулі для управління доступом, які дозволяють більш гнучкий контроль над доступом до роутів та інших елементів Drupal.

У Drupal 8, ви можете використовувати динамічні права доступу для керування доступом до роутів на основі певних умов чи параметрів. Це дозволяє вам робити більш гнучкі та контекстуальні рішення щодо доступу до функціональності вашого модуля.

Для використання динамічних прав доступу до роутів, ви можете визначити власний сервіс, який реалізує інтерфейс `AccessInterface` та надає метод `access()`. Цей метод повинен повернути `AccessResult` об'єкт, який визначає, чи має користувач доступ до роуту.

Ось приклад створення динамічних прав доступу:

Створіть новий сервіс, який реалізує `AccessInterface`:

```
namespace Drupal\my_module\Access;

use Drupal\Core\Access\AccessResult;
use Drupal\Core\Routing\Access\AccessInterface;
use Drupal\Core\Session\AccountInterface;

class MyAccessCheck implements AccessInterface {

  public function access(AccountInterface $account) {
    // Ваш код для перевірки прав доступу на основі умов.
    // Поверніть AccessResult з відповідними значеннями.
    // Наприклад:
    if ($account->hasPermission('access content')) {
      return AccessResult::allowed();
    }
    return AccessResult::forbidden();
  }
}
```

Зареєструйте сервіс у файлі `my_module.services.yml` вашого модуля:

```
services:
  my_module.access_check:
    class: Drupal\my_module\Access\MyAccessCheck
    tags:
      - { name: access_check, applies_to: _route:my_module.my_route }
```

Визначте роут вашого модуля і вкажіть динамічні права доступу. В цьому прикладі, `MyAccessCheck` сервіс викликається для перевірки прав доступу до роуту `my_module.my_route`. Залежно від умов у методі `access()`, буде повернуто відповідне значення `AccessResult`, що визначає, чи має користувач доступ до роуту.

```
my_module.my_route:
  path: '/my-path'
  defaults:
    _controller: '\Drupal\my_module\Controller\MyController::myMethod'
    _title: 'My Page'
  requirements:
    _access: 'my_module.access_check:access'
```

Ви можете налаштувати свої власні умови та логіку в методі `access()`, щоб реалізувати динамічний контроль доступу до роутів у вашому модулі.

При створенні роутів в Drupal 8, важливо забезпечити належний рівень безпеки, щоб захистити вашу програму від потенційних атак. Ось деякі ключові аспекти безпеки, на які варто звернути увагу при створенні роутів:

- Перевірка прав доступу: Визначте належні права доступу до роуту, щоб лише авторизовані користувачі з необхідними дозволами мали доступ до функціональності, яку надає роут. Використовуйте ключ `_permission` або створюйте власні методи перевірки доступу.
- Захист від XSS-атак: Переконайтеся, що дані, які передаються через роут, відфільтровані та екрановані, щоб уникнути виконання вредонебезпечного коду на стороні клієнта. Використовуйте функції, такі як `Html::escape()` або `FilterFormat::create()` для забезпечення безпечного відображення даних.
- Захист від SQL-ін'єкцій: Уникайте безпечних ризиків SQL-ін'єкцій, використовуючи параметризовані запити та валідацію вхідних даних. Використовуйте методи `query()` або `select()` з використанням зв'язаних параметрів.
- Захист від CSRF-атак: Використовуйте механізми захисту від подібних атак, такі як `Form API` або генерація токенів CSRF для перевірки справжності відправлених форм.
- Авторизація і аутентифікація: Впевніться, що ви користуєтеся правильними механізмами авторизації і аутентифікації, такими як `Drupal\Core\Session\AccountInterface`, для перевірки даних користувача та управління доступом до функціональності.
- Перевірка валідності даних: Перевіряйте та валідуйте всі вхідні дані, щоб

- уникнути вразливостей, таких як переповнення буфера, втручання в SQL-запити тощо. Використовуйте методи валідації, такі як `EntityValidationConstraint` або `FormValidatorInterface`, для перевірки правильності вхідних даних.
- Логування та моніторинг: Включіть логування подій і помилок, щоб виявити та відстежувати можливі проблеми з безпекою. Використовуйте інструменти, такі як `watchdog()` або стандартний логгер для записування логів.

Пам'ятайте, що безпека - це процес, і вона вимагає постійного оновлення та покращення. Регулярно оновлюйте ваш Drupal 8, використовуйте надійні модулі безпеки та слідкуйте за рекомендаціями Drupal Security Team.

3.2.4. КОНТРОЛЛЕРИ

Контролери є важливою складовою частиною архітектури в Drupal 8. Вони відповідають за обробку запитів і виконання необхідних дій, таких як виведення вмісту, обробка форм, взаємодія з базою даних тощо. Розробка контролерів в Drupal 8 відбувається з використанням паттерну проектування "Контролер Front Controller" і основується на Symfony Framework.

Основні принципи та структура контролерів в Drupal 8:

Клас контролера: Контролери в Drupal 8 є класами PHP, які реалізують інтерфейс `ControllerInterface` або успадковуються від базового класу `ControllerBase`. Клас контролера містить методи, що відповідають за обробку різних дій.

Методи контролера: Контролери мають різні методи для обробки різних типів запитів. Основним методом є `public function __invoke()` або `public function execute()`, який викликається при обробці запиту. Інші методи можуть бути викликані в залежності від типу запиту або дії.

Маршрутизація: Контролери пов'язуються з маршрутами, які визначають URL-шляхи до контролера. Маршрути в Drupal 8 визначаються у файлі `my_module.routing.yml` вашого модуля. Кожна маршрута має шлях, назву контролера, дія і, за необхідності, параметри.

Залежності та сервіси: Контролери можуть використовувати сервіси, щоб отримати доступ до додаткової функціональності. Вони можуть використовувати ін'єкцію залежностей для отримання доступу до необхідних сервісів.

Відповіді: Контролери повертають відповіді відповідно до типу запиту і дії. Відповіді можуть бути HTML-сторінками, JSON-даними, файлами тощо. Використовуйте класи відповідей, такі як `Response`, `JsonResponse`, `FileResponse` для повернення відповідей.

Кешування: Контролери можуть використовувати кешування для збереження попередньо оброблених результатів. Використовуйте анотації, такі як `@Cacheable`, `@CacheTags`, `@CacheContexts` для конфігурації кешування.

Тестування: Важливим аспектом розробки контролерів є тестування, щоб переконатися, що вони працюють належним чином. Використовуйте тести, такі як Unit-тести, Kernel-тести або Browser-тести, для перевірки функціональності контролерів.

Контролери в Drupal 8 надають широкі можливості для розробки динамічних та масштабованих додатків. Вони дозволяють ефективно обробляти запити і забезпечувати необхідний функціонал.

Щоб створити контролер в модулі Drupal 8, слід виконати наступні кроки:

Створіть новий файл контролера у папці src/Controller модуля з розширенням `.php`, наприклад `MyModuleController.php`.

Відкрийте цей файл і визначте новий клас контролера. Назва класу повинна відповідати назві файлу та використовувати простір імен вашого модуля. Наприклад:

```
namespace Drupal\my_module\Controller;

use Drupal\Core\Controller\ControllerBase;

class MyModuleController extends ControllerBase {
  // Код контролера тут.
}
```

Внутрішній код контролера повинен містити методи для обробки різних дій. Наприклад, метод `hello()` виведе просте повідомлення:

```
public function hello() {
  return [
    '#markup' => $this->t('Hello, world!'),
  ];
}
```

Визначте маршрути для контролера. У вашому модулі створіть файл `my_module.routing.yml` і вкажіть шляхи до відповідних методів контролера:

```
my_module.hello:
  path: '/my-module/hello'
  defaults:
    _controller: '\Drupal\my_module\Controller\MyModuleController::hello'
    _title: 'Hello'
  requirements:
    _permission: 'access content'
```

Після збереження файлів очистіть кеш Drupal, щоб система розпізнала нові маршрути:

```
drush cache:rebuild
```

Тепер ви можете отримати доступ до свого контролера за URL-адресою `/my-module/hello` і побачити виведене повідомлення "Hello, world!".

Не забудьте вказати правильні простори імен для вашого модуля та назви файлів і методів контролера відповідно до ваших потреб.

При створенні контролера в Drupal 8 варто звернути увагу на наступні аспекти:

Простір імен: Використовуйте правильні простори імен для класів контролерів. Простір імен повинен відповідати структурі папок модуля, в якому ви створюєте контролер.

Наслідування: Контролери повинні успадковуватись від класу `ControllerBase`. Це забезпечить доступ до необхідних функцій та методів для обробки запитів.

Методи контролера: Контролери повинні містити методи для обробки різних дій. Наприклад, метод `hello()` може бути викликаний при обробці URL-адреси `/hello` і відповідати за виведення потрібного вмісту.

Маршрутизація: Визначте маршрути для контролера, щоб вказати шляхи, за якими він буде доступний. Переконайтеся, що маршрути відповідають практикам розробки Drupal і мають необхідні вимоги, як-то права доступу або параметри.

Сервіси та залежності: Контролери можуть використовувати сервіси для отримання доступу до додаткової функціональності. Використовуйте ін'єкцію залежностей для отримання необхідних сервісів.

Тестування: Не забувайте про тестування контролерів, щоб переконатися, що вони працюють належним чином. Використовуйте тести, такі як Unit-тести, Kernel-тести або Browser-тести, для перевірки функціональності контролерів.

Кешування: При обробці запитів контролери можуть використовувати кешування для збереження попередньо оброблених результатів. Використовуйте анотації, такі як `@Cacheable`, `@CacheTags`, `@CacheContexts`, для конфігурації кешування.

Безпека: Забезпечте безпеку контролерів, перевіривши права доступу до відповідних маршрутів і обмеживши доступ до конфіденційної інформації або небезпечних операцій.

Загалом, при створенні контролерів в Drupal 8 важливо дотримуватись стандартів і практик розробки, забезпечувати безпеку, тестувати функціональність і дотримуватись засад ООП.

В Drupal 8 колбеки контролерів можуть використовувати кешування для збереження попередньо оброблених результатів і покращення продуктивності. Це досягається за допомогою анотацій та методів для конфігурації кешування.

Основні кроки для налаштування кешування у колбеках контролера такі:

Визначте анотацію `@Cacheable` для методу колбека, що підлягає кешуванню. Це повідомляє Drupal, що результати методу можуть бути кешовані.

```
use Drupal\Core\Cache\CacheableResponseInterface;
use Drupal\Core\Cache\CacheableResponseTrait;
```

```
/**
 * @Cacheable
 */
public function myCallback() {
  // Ваш код тут.
}
```

Реалізуйте інтерфейс `CacheableResponseInterface` та включіть його в контролер, а також використовуйте трейт `CacheableResponseTrait`. Це дозволить контролеру повернути кешований відповідь з належними заголовками кешування.

```
use Drupal\Core\Cache\CacheableResponseInterface;
use Drupal\Core\Cache\CacheableResponseTrait;
use Symfony\Component\HttpFoundation\Response;

class MyController implements CacheableResponseInterface {
  use CacheableResponseTrait;

  public function myCallback() {
    // Ваш код тут.
    $response = new Response('Hello, world!');
    $this->addCacheableDependency($response);
    return $response;
  }
}
```

Використовуйте метод `addCacheableDependency()`, щоб додати залежності кешування до відповіді. Наприклад, ви можете вказати залежність від кешу блоків, налаштувань, ролей користувачів тощо.

```
$this->addCacheableDependency($response, $cacheableDependency);
```

Використовуйте анотації `@CacheContexts`, `@CacheTags`, `@CacheMaxAge` і `@CacheableMetadata` для подальшої настройки кешування, вказуючи контексти кешування, теги, максимальний вік кешу та метадані кешу.

```
/**
 * @Cacheable
 * @CacheContexts({"user", "language"})
 * @CacheTags({"my_custom_cache_tag"})
 * @CacheMaxAge(3600)
  *  @CacheableMetadata(key="my_custom_cache_key",  cache_contexts={"user"}, cache_tags={"my_custom_cache_tag"})
 */
public function myCallback() {
 // Ваш код тут.
}
```

Використання кешування у колбеках контролера допоможе зменшити час відповіді та навантаження на сервер, особливо для сторінок, які рідко змінюються або мають великий обчислювальний витрати. Враховуйте, що кешування може бути чутливим до змін, тому слід ретельно настроювати залежності та правильно оновлювати кеш при змінах в даних.

3.2.5. ПЛАГІНИ

Плагіни в Drupal 8 - це розширювальні модулі, які надають змогу додавати нові функціональність, розширювати функціональність ядра Drupal або інших модулів. Вони дозволяють розробникам реалізовувати власні логіки та розширювати систему без змін вихідного коду ядра.

Основні характеристики плагінів в Drupal 8:

Типи плагінів: Drupal 8 надає кілька типів плагінів, таких як блоки, полі, виділення тексту, валидатори форм, роутери, аутентифікація та багато інших. Кожен тип плагіна визначає свої правила та інтерфейси, які розробники можуть використовувати для створення своїх власних плагінів.

Анотації: Плагіни в Drupal 8 використовують анотації для визначення своїх властивостей та конфігурації. Анотації дозволяють визначати імена, типи, описи та інші важливі атрибути плагінів.

Плагін-менеджери: Drupal 8 надає плагін-менеджери, які допомагають завантажувати, встановлювати та використовувати плагіни. Плагін-менеджери автоматично сканують папки модулів для знаходження плагінів, завантажують їх та реєструють для використання.

Розширення та перевизначення: Плагіни в Drupal 8 можуть бути розширеними та перевизначеними. Це означає, що інші модулі можуть додавати нові варіанти плагінів або замінювати варіанти плагінів, визначені в інших модулях. Це дозволяє розширювати функціональність плагінів та змінювати їх поведінку без втручання в код плагінів.

Конфігурація: Багато плагінів в Drupal 8 мають конфігураційні параметри, які можна настроїти через адміністративний інтерфейс. Це дозволяє адміністраторам налаштовувати плагіни без необхідності втручання в код.

Основні кроки для створення плагіна в Drupal 8:

- Створіть папку модуля або виберіть існуючу папку для розміщення плагіна.
- Створіть файл плагіна з відповідними анотаціями та методами.
- Реалізуйте необхідні інтерфейси та методи для роботи плагіна.
- Зареєструйте плагін у плагін-менеджері, щоб він був доступний для використання.
- Налаштуйте конфігураційні параметри плагіна, якщо необхідно.
- Використовуйте плагін у своєму коді або інтегруйте його з іншими модулями.

Це загальний опис процесу створення плагіна в Drupal 8. Залежно від типу плагіна і його призначення можуть бути додаткові кроки та особливості.

Розглянемо приклад створення плагіну через створення кастомного блоку в Drupal 8:

Створіть папку для вашого модуля в папці `modules` вашого Drupal-сайту. Дайте йому ім'я, наприклад `custom_block`. У папці модуля створіть файл `custom_block.info.yml` з таким вмістом:

```
name: 'Custom Block'
type: module
description: 'Creates a custom block.'
core_version_requirement: ^8 || ^9
package: Custom
dependencies:
  - block
```

У папці модуля створіть папку `src/Plugin/Block` і створіть файл `CustomBlock.php`. Відкрийте файл і введіть наступний код:

```
namespace Drupal\custom_block\Plugin\Block;
use Drupal\Core\Block\BlockBase;

/**
 * Provides a custom block.
 *
 * @Block(
 *   id = "custom_block",
 *   admin_label = @Translation("Custom Block"),
 *   category = @Translation("Custom")
 * )
 */
class CustomBlock extends BlockBase {

  /**
   * {@inheritdoc}
   */
  public function build() {
    return [
      '#markup' => $this->t('Hello, world!'),
    ];
  }
}
```

Код вище описує клас `CustomBlock`, який розширює `BlockBase`. У методі `build()` ми повертаємо масив з ключем `#markup`, який містить привітання "Hello, world!".

У папці модуля створіть файл `custom_block.module` і введіть наступний код:

```
use Drupal\Core\Routing\RouteMatchInterface;

/**
 * Implements hook_theme().
 */
function custom_block_theme() {
  return [
    'custom_block_template' => [
      'variables' => [
        'content' => NULL,
      ],
      'template' => 'custom-block',
    ],
  ];
}

/**
 * Implements hook_block_info().
 */
function custom_block_block_info() {
  $blocks = [];

  $blocks['custom_block'] = [
    'info' => t('Custom Block'),
    'cache' => DRUPAL_NO_CACHE,
  ];

  return $blocks;
}
```

```
/**
 * Implements hook_block_view().
 */
function custom_block_block_view($delta = '') {
  $block = [];

  if ($delta === 'custom_block') {
    $content = [
      '#theme' => 'custom_block_template',
      '#content' => 'Hello, world!',
    ];

    $block['content'] = $content;
  }

  return $block;
}
```

Код вище визначає кілька хук-функцій. `custom_block_theme()` визначає шаблон `custom-block.html.twig`, який буде використовуватись для відображення блоку. `custom_block_block_info()` визначає інформацію про блок, а `custom_block_block_view()` визначає вміст блоку.

У папці модуля створіть папку `templates` і створіть файл `custom-block.html.twig` з таким вмістом:

```
<div class="custom-block">
  {{ content }}
</div>
```

Цей шаблон визначає, як саме буде відображатись блок. В даному прикладі ми просто виводимо змінну `{{ content }}`, яка містить вміст блоку.

Після створення вашого блоку оновіть кеш Drupal. Це можна зробити через адміністративний інтерфейс Drupal або використовуючи команду `drush cr` в терміналі.

Зайдіть в адміністративний інтерфейс Drupal і перейдіть на сторінку "Розміщення блоків" (`/admin/structure/block`). Знайдіть ваш блок за назвою "Custom Block" у списку доступних блоків і розмістіть його в потрібному регіоні на вашому сайті.

Іншим прикладом плагінів є віджети та форматери. Розглянемо їх створення.

Звичайно! Ось детальна інструкція з прикладами коду для створення кастомного віджета та форматера в Drupal 8:

Створення кастомного віджета:

Крок 1: Створення модуля

Створіть папку для вашого модуля в папці `modules` вашого Drupal-сайту. Дайте йому ім'я, наприклад `custom_widgets`. У папці модуля створіть файл `custom_widgets.info.yml` з таким вмістом:

```
name: 'Custom Widgets'
type: module
description: 'Creates custom widgets and formatters.'
core_version_requirement: ^8 || ^9
package: Custom
```

У папці модуля створіть папку `src/Plugin/Field/Widget` і створіть файл `CustomWidget.php`. Відкрийте файл і введіть наступний код:

```
namespace Drupal\custom_widgets\Plugin\Field\Widget;

use Drupal\Core\Field\FieldItemListInterface;
use Drupal\Core\Field\WidgetBase;
use Drupal\Core\Form\FormStateInterface;
```

```
/**
 * Plugin implementation of the 'custom_widget' widget.
 *
 * @FieldWidget(
 *   id = "custom_widget",
 *   label = @Translation("Custom Widget"),
 *   field_types = {
 *     "text"
 *   }
 * )
 */
class CustomWidget extends WidgetBase {

  /**
   * {@inheritdoc}
   */
  public function formElement(FieldItemListInterface $items, $delta, Array $element, Array &$form, FormStateInterface $form_state) {
    $value = isset($items[$delta]->value) ? $items[$delta]->value : '';

    $element['value'] = [
      '#type' => 'textfield',
      '#title' => t('Custom Field'),
      '#default_value' => $value,
    ];

    return $element;
  }
}
```

У цьому прикладі створюється клас `CustomWidget`, який розширює `WidgetBase`. У методі `formElement()` ми повертаємо елемент форми для відображення на сторінці редагування вмісту. У нашому випадку, ми створюємо текстове поле з назвою "Custom Field".

У папці модуля створіть файл `custom_widgets.module` і введіть наступний код:

```
/**
 * @file
 * Custom widgets module.
 */

use Drupal\Core\Routing\RouteMatchInterface;

/**
 * Implements hook_field_widget_info().
 */
function custom_widgets_field_widget_info() {
  return [
    'custom_widget' => [
      'label' => t('Custom Widget'),
      'field types' => ['text'],
    ],
  ];
}
```

Цей код визначає віджет `custom_widget` за допомогою `hook_field_widget_info()`. У нашому випадку, ми встановлюємо цей віджет для поля типу `text`.

Створення кастомного форматера:

У папці модуля створіть папку `src/Plugin/Field/FieldFormatter` і створіть файл `CustomFormatter.php`. Відкрийте файл і введіть наступний код:

```
namespace Drupal\custom_widgets\Plugin\Field\FieldFormatter;
use Drupal\Core\Field\FieldItemListInterface;
use Drupal\Core\Field\FormatterBase;
use Drupal\Core\Field\Plugin\Field\FieldFormatter\StringFormatter;

/**
 * Plugin implementation of the 'custom_formatter' formatter.
 *
 * @FieldFormatter(
 *   id = "custom_formatter",
 *   label = @Translation("Custom Formatter"),
 *   field_types = {
 *     "text"
 *   }
 * )
 */
class CustomFormatter extends StringFormatter {

  /**
   * {@inheritdoc}
   */
  public function viewElements(FieldItemListInterface $items, $langcode) {
    $elements = parent::viewElements($items, $langcode);

    foreach ($elements as &$element) {
      $element['#markup'] = 'Custom formatted value: ' . $element['#markup'];
    }

    return $elements;
  }
}
```

У цьому прикладі створюється клас `CustomFormatter`, який розширює `StringFormatter`. У методі `viewElements()` ми модифікуємо вміст елементів поля шляхом додавання префіксу "Custom formatted value".

У файлі `custom_widgets.module` вже створеному раніше, додайте наступний код:

```
/**
 * Implements hook_field_formatter_info().
 */
function custom_widgets_field_formatter_info() {
  return [
    'custom_formatter' => [
      'label' => t('Custom Formatter'),
      'field types' => ['text'],
    ],
  ];
}
```

Цей код визначає форматер `custom_formatter` за допомогою `hook_field_formatter_info()`. У нашому випадку, ми встановлюємо цей форматер для поля типу `text`.

Після створення вашого віджета та форматера оновіть кеш Drupal. Це можна зробити через адміністративний інтерфейс Drupal або використовуючи команду `drush cr` в терміналі.

Зауваження:

Переконайтеся, що ваш модуль `custom_widgets` увімкнений в адміністративному інтерфейсі Drupal перед тим, як ви спробуєте використовувати ваш кастомний віджет або форматер.

Вам також знадобиться встановлений і активований модуль `field_ui`, який дозволяє керувати налаштуваннями полів через адміністративний інтерфейс.

При створенні плагінів у Drupal 8 є кілька важливих аспектів, на які варто звертати увагу. Ось деякі з них:

Правильне розміщення файлів: Файли плагінів повинні бути розміщені у відповідних папках модуля. Наприклад, плагіни блоків повинні бути розміщені в папці `src/Plugin/Block`, а плагіни полів - у папці `src/Plugin/Field`.

Коректне ім'я класу: Клас плагіна повинен мати коректне ім'я згідно з конвенціями Drupal. Ім'я класу повинно починатися з префіксу, який відповідає назві вашого модуля, а потім слідує назва папки плагіну та назва самого плагіну. Наприклад, якщо ваш модуль називається `custom_module`, а плагін - `CustomPlugin`, ім'я класу буде `CustomModuleCustomPlugin`.

Використання анотацій: Кожен плагін повинен мати анотацію, що описує його. У цій анотації ви повинні вказати ID плагіна, мітку (label) та типи полів або інших компонентів, для яких плагін призначений.

Використання базових класів: Drupal 8 надає базові класи для різних типів плагінів, які ви можете розширити. Наприклад, для плагінів блоків ви можете розширити клас `BlockBase`, а для плагінів полів - клас `FieldFormatterBase` або `FieldWidgetBase`. Це забезпечує консистентність та допомагає вам використовувати вбудовані функції та методи, що спрощують роботу з плагінами.

Валідація та санітізація даних: Переконайтеся, що ви правильно валідуєте та санітізуєте будь-які дані, які вводяться або виводяться через ваш плагін. Це допоможе запобігти атакам на безпеку та некоректному відображенню даних.

Тестування: Завжди тестуйте ваші плагіни, щоб переконатися, що вони працюють коректно і не порушують функціональність сайту. Використовуйте автоматизовані тести (наприклад, PHPUnit) для перевірки різних аспектів роботи плагінів.

Документація: Додайте коментарі до коду вашого плагіна, пояснюючи його функціональність та особливості. Це допоможе іншим розробникам легше зрозуміти ваш код та працювати з ним.

Ці аспекти є лише загальними рекомендаціями, і на практиці можуть бути інші нюанси, залежно від конкретної ситуації та потреб вашого проекту. Проте, враховуючи ці рекомендації, ви зможете створити якісні та добре структуровані плагіни у вашому Drupal 8 проекті.

3.2.6. ФОРМИ

У Drupal 8 для створення форм в модулі використовується система форм (Form API), яка надає потужні інструменти для створення, валідації та обробки форм.

Розглянемо створення форми в модулі Drupal 8:

Створіть клас форми: Створіть новий клас форми у вашому модулі. Цей клас повинен успадковувати `FormBase` або інший базовий клас, що відповідає типу форми, яку ви хочете створити.

```
use Drupal\Core\Form\FormBase;
use Drupal\Core\Form\FormStateInterface;

class MyForm extends FormBase {
  // Реалізуйте методи класу тут
}
```

Реалізуйте метод `getFormId()`: У вашому класі форми реалізуйте метод `getFormId()`, який повертає унікальний ідентифікатор форми.

```
public function getFormId() {
  return 'my_form';
}
```

Реалізуйте метод `buildForm()`: У методі `buildForm()` ви описуєте структуру форми та її елементи, використовуючи методи з об'єкта `$form`. Ви також можете додати обробники подій (колбеки), які будуть виконуватись при поданні форми.

Форма міститиме елементи які будуть виведені на формі - рендер та форм елементи, які ми розглянемо дещо швидше.

Форма може містити довільну кількість елементів, як статичних, так і динамічних - з використанням асинхронної валідаці з використанням Ajax.

Всі елементи форми в подальшому можуть бути відрендерені як стандартним шляхом так і з використанням кастомізованого темплейта.

```
public function buildForm(array $form, FormStateInterface $form_state) {
  $form['name'] = [
    '#type' => 'textfield',
    '#title' => $this->t('Name'),
    '#required' => TRUE,
  ];

  $form['submit'] = [
    '#type' => 'submit',
    '#value' => $this->t('Submit'),
  ];

  return $form;
}
```

Реалізуйте метод `submitForm()`: У методі `submitForm()` ви описуєте логіку обробки даних, які були введені в форму при поданні.

```
public function submitForm(array &$form, FormStateInterface $form_state) {
  $name = $form_state->getValue('name');
  // Обробка введених даних форми
}
```

Налаштуйте маршрут для форми: В файлі `mymodule.routing.yml` вашого модуля визначте маршрут, який пов'язаний з вашою формою.

```
myform:
  path: '/myform'
  defaults:
    _form: '\Drupal\mymodule\Form\MyForm'
    _title: 'My Form'
  requirements:
    _permission: 'access content'
```

Для виведення форми на сторінці використовуйте шаблони або блоки. За допомогою методу `create()` сервісу `FormBuilder` можна створити екземпляр вашої форми та отримати HTML-представлення форми для вставки на сторінку.

```
$form = \Drupal::formBuilder()->getForm('\Drupal\mymodule\Form\MyForm');
$html = \Drupal::service('renderer')->render($form);
```

Це загальний опис процесу створення форми в модулі Drupal 8. Ви можете додатково налаштувати валідацію, обробники подій та інші функціональні можливості, які надає система форм (Form API) Drupal 8.

У Drupal 8 форми будуються з використанням форм елементів (Form API elements), які надають різні типи полів, кнопок, вибірок, радіо-кнопок, прапорців та багато іншого. Форм елементи використовуються для створення різноманітних полів введення та відображення даних у формах модулів Drupal 8.

Ось деякі приклади найпоширеніших форм елементів та їх використання у формах модулів Drupal 8:

Textfield (текстове поле):

```
$form['name'] = [
  '#type' => 'textfield',
  '#title' => 'Name',
  '#required' => TRUE,
];
```

Select (випадаючий список):

```
$form['category'] = [
  '#type' => 'select',
  '#title' => 'Category',
  '#options' => ['option1' => 'Option 1', 'option2' => 'Option 2'],
];
```

Checkbox (прапорець):

```
$form['agree'] = [
  '#type' => 'checkbox',
  '#title' => 'I agree to the terms and conditions',
];
```

Radios (радіо-кнопки):

```
$form['color'] = [
  '#type' => 'radios',
  '#title' => 'Color',
  '#options' => ['red' => 'Red', 'blue' => 'Blue', 'green' => 'Green'],
];
```

Це лише деякі приклади форм елементів, які можна використовувати у формах модулів Drupal 8. Існує багато інших типів полів та елементів, таких як кнопки, файлові завантаження, дата, час, телефон і т.д. Ви можете вибрати відповідний форм елемент залежно від типу даних, які ви хочете отримати від користувача.

Крім того, ви можете змінювати зовнішній вигляд елементів форми за допомогою рендер елементів (Render API elements). Рендер елементи дозволяють визначити, як елементи форми повинні бути відображені, додавати CSS класи, атрибути та інші налаштування для оформлення.

Наприклад, для додавання CSS класу до текстового поля можна використати такий код:

```
$form['name'] = [
  '#type' => 'textfield',
   '#title' => 'Name',
  '#required' => TRUE,
  '#attributes' => [
    'class' => ['my-custom-class'],
  ],
];
```

Таким чином, ви можете створювати форми з використанням різних форм елементів та налаштовувати їх зовнішній вигляд за допомогою рендер елементів у формах модулів Drupal 8.

При створенні форм в модулі Drupal варто звертати увагу на такі моменти:

Валідація даних: Перевірте коректність введених даних і забезпечте належну валідацію на серверному боці. Використовуйте вбудовані методи перевірки даних, такі як `Required`, `Length`, `Email`, або створюйте власні правила валідації.

Захист від атак: Застосовуйте належні заходи безпеки, такі як захист від CSRF-атак (Cross-Site Request Forgery) та XSS-атак (Cross-Site Scripting). Використовуйте вбудовані методи безпеки, наприклад, додавання токенів або очищення введених даних.

Правильна обробка даних: Ретельно обробляйте введені дані, перевіряйте їх на валідність та наявність. Уникайте SQL-ін’єкцій і інших вразливостей. Використовуйте функції Drupal API, такі як `db_insert()`, `db_update()` або `db_query()` для безпечного взаємодії з базою даних.

Коректне відображення даних: Зверніть увагу на належний відображення даних у формах. Використовуйте вбудовані форматери, наприклад, `Drupal\Core\Field\FieldFormatterInterface` або створюйте власні форматери для правильного відображення даних.

Локалізація: Передбачте можливість локалізації форм, зокрема переклади текстів, форматування чисел та дат, з урахуванням мовних налаштувань Drupal.

Розділення логіки: Розділіть логіку обробки форми від представлення форми. Використовуйте контролери та сервіси для обробки даних, а шаблони для відображення форми.

Використання Drupal API: Використовуйте доступні функції та методи Drupal API для спрощення розробки форм. Наприклад, використовуйте методи Form API, або сервіси, такі як `entity.manager` для доступу до сутностей.

Надійність та продуктивність: Забезпечте оптимальну продуктивність форм шляхом оптимізації запитів до бази даних, кешування та використання правильних методів роботи з даними.

Загалом, створення форм в модулі Drupal вимагає уваги до безпеки, правильної обробки даних та збереження стандартів Drupal.

3.2.7. СЕРВІСИ

Сервіси є основними компонентами архітектури Drupal 8 і використовуються для організації перевикористовування коду, виконання бізнес-логіки та забезпечення розширюваності системи. Вони забезпечують відокремлення логіки від представлення і забезпечують ін'єкцію залежностей, що полегшує тестування та підтримку коду.

Основні особливості та принципи сервісів в Drupal 8:

Ін'єкція залежностей: Сервіси залежать від інших сервісів, які вони отримують за допомогою ін'єкції залежностей. Це дозволяє забезпечити модульність, ослабити зв'язаність та спростити тестування.

Конфігуровані сервіси: Drupal 8 має систему конфігураційних сервісів, що дозволяє налаштувати параметри сервісів через файл конфігурації. Це дозволяє змінювати поведінку сервісів без необхідності змінювати код.

Життєвий цикл сервісу: Сервіси можуть мати життєвий цикл, що означає, що вони можуть бути створені, знищені та оновлені в певний момент часу. Це дозволяє керувати ресурсами та пам'яттю, використовуваними сервісами.

Реалізація через класи: Сервіси в Drupal 8 зазвичай реалізовані через класи. Класи описують функціонал та методи сервісу, а також забезпечують реалізацію інтерфейсів.

Сервісний контейнер: Drupal 8 використовує сервісний контейнер для керування та реєстрації сервісів. Сервісний контейнер є центральним механізмом для отримання доступу до сервісів у всій системі.

Створення та використання сервісів в Drupal 8 включає наступні кроки:

Створення класу сервісу: Створіть клас, який реалізує необхідну функціональність та інтерфейси.

Реєстрація сервісу: Зареєструйте свій сервіс у сервісному контейнері, використовуючи відповідні механізми, такі як файл services.yml.

Ін'єкція залежностей: Ін'єкція залежностей дає можливість отримати доступ до інших сервісів, які використовуються в сервісі. Ін'єкція залежностей може бути здійснена через конструктор, методи або через анотації.

Використання сервісу: Після реєстрації та ін'єкції залежностей сервіс готовий до використання у ваших модулях, темах або інших компонентах Drupal.

Звертайте увагу на правильне налаштування та використання сервісів, використання інтерфейсів для забезпечення гнучкості та розширюваності, а також враховуйте безпекові аспекти, які можуть виникати при роботі з сервісами в Drupal 8.

Розглянемо приклад створення класу сервісу: Давайте припустимо, що ваш модуль називається "MyModule", а клас сервісу - "MyService". Створіть файл "MyService.php" у вашому модулі з таким вмістом:

```
namespace Drupal\my_module;
use Drupal\Core\Messenger\MessengerInterface;

class MyService {

 protected $messenger;

 public function __construct(MessengerInterface $messenger) {
  $this->messenger = $messenger;
 }

 public function showMessage($message) {
  $this->messenger->addMessage($message);
 }
}
```

Реєстрація сервісу: Створіть файл "my_module.services.yml" у вашому модулі та додайте наступний код:

```
services:
 my_module.my_service:
  class: Drupal\my_module\MyService
  arguments: ['@messenger']
```

Використання сервісу: Ви можете використовувати свій сервіс у інших частинах вашого модуля або в інших модулях Drupal 8, ін'єкцію сервісу можна здійснити за допомогою конструктора або через методи.

Наприклад, у вашому контролері можна використати сервіс:

```
namespace Drupal\my_module\Controller;

use Drupal\Core\Controller\ControllerBase;
use Symfony\Component\DependencyInjection\ContainerInterface;

class MyController extends ControllerBase {

 protected $myService;

 public static function create(ContainerInterface $container) {
  return new static(
   $container->get('my_module.my_service')
  );
 }

 public function __construct(MyService $myService) {
  $this->myService = $myService;
 }

 public function myPage() {
  $this->myService->showMessage('Hello, Drupal 8!');
  // Інші дії тут...
 }
}
```

Таким чином, ви створили свій власний сервіс та використали його в контролері для виконання певних дій.

Використання сервісів в модулях Drupal 8 має свої переваги та недоліки. Ось деякі з них:

Переваги:

Розподілення відповідальностей: Використання сервісів дозволяє розділити логіку програми на окремі компоненти, що полегшує розробку та підтримку коду.

Перевикористання коду: Сервіси можна використовувати в різних частинах модуля або навіть в інших модулях Drupal 8, що сприяє перевикористанню коду та зменшенню дублювання.

Тестування: Використання сервісів полегшує тестування, оскільки окремі компоненти можна тестувати окремо без необхідності створювати складні залежності.

Недоліки:

Складність: Розуміння та налаштування сервісів може бути складним для новачків у Drupal 8. Потрібно мати розуміння про ін'єкцію залежностей та конфігурацію сервісів.

Зависимість від контейнера: Сервіси залежать від контейнера впровадження (dependency injection container) в Drupal 8. Це може вплинути на продуктивність та зрозумілість коду.

Перевантаження сервісів: Надмірне використання сервісів та неправильне проектування можуть призвести до перевантаження кодової бази та збільшення складності.

Враховуючи ці переваги та недоліки, важливо ретельно планувати та проектувати використання сервісів у вашому модулі, забезпечуючи баланс між розширюваністю, зрозумілістю та ефективністю коду.

У розробці на Drupal 8 можна використовувати певні підходи для розділення коду між формами, контролерами і сервісами, а також для розділення на бізнес-логіку та системну логіку. Ось кілька порад, як це можна зробити:

Розділення коду між формами і контролерами:

Форми в Drupal 8 зазвичай використовуються для збору даних від користувача. Рекомендовано розміщувати код форми в окремих класах форм. Вони можуть бути розташовані в папці `src/Form` вашого модуля.

Контролери використовуються для обробки запитів і рендерингу відповідних сторінок. Рекомендовано розміщувати код контролера в окремих класах контролера. Вони можуть бути розташовані в папці `src/Controller` вашого модуля.

Розділення коду між контролерами і сервісами:

Контролери в Drupal 8 відповідають за обробку запитів, а сервіси використовуються для виконання певних операцій або бізнес-логіки. Рекомендовано виконувати бізнес-логіку в окремих сервісних класах і використовувати їх у контролерах.

Розташуйте сервіси в папці `src/Service` вашого модуля. Код сервісів може включати методи для виконання певних операцій або реалізацію бізнес-логіки.

Розділення на бізнес-логіку та системну логіку:

Бізнес-логіка відповідає за операції, специфічні для вашої доменної області або функціональності. Цей код повинен бути розміщений в сервісах або інших класах, які відповідають за бізнес-логіку вашого модуля.

Системна логіка відповідає за загальну функціональність Drupal, таку як маршрутизація, доступ до бази даних або обробка подій. Цей код може бути розташований в контролерах, хуках або інших системних класах.

Загалом, важливо розуміти, що це лише рекомендації, а не жорсткі правила. Ви можете організувати свій код у відповідності до потреб вашого проекту, з урахуванням принципів чистої архітектури та добрих практик програмування.

3.2.7. ПРАВА ДОСТУПУ

У Drupal 8 для оголошення і використання прав доступу в модулі використовується система ролей та дозволів, що вбудована у ядро Drupal. Основні кроки для оголошення і використання прав доступу в модулі виглядають наступним чином:

Оголошення прав доступу:

- Створіть файл `my_module.permissions.yml` в корені вашого модуля.
- У цьому файлі визначте права доступу, які будуть доступні вашому модулю.

Наприклад:

```
access content:
  title: 'Access content'
  description: 'Access the content created by the module'
```

Використання прав доступу:

Для використання оголошених прав доступу у вашому модулі використовуйте хуки або анотації, залежно від ваших потреб.

Наприклад, якщо ви хочете перевірити право доступу до певної функціональності в вашому контролері, ви можете використовувати наступний приклад коду:

```
if ($this->currentUser->hasPermission('access content')) {
  // Робіть те, що доступне користувачам з правом доступу "Access content".
}
```

Відображення прав доступу у формах адміністративного інтерфейсу:

Якщо ви хочете відобразити оголошені права доступу у формах адміністративного інтерфейсу, ви можете використовувати `hook_form_alter()` для додавання полів вибору дозволів або перевіряти права доступу у функціях зворотного виклику форми.

Налаштування прав доступу для ролей:

Після оголошення прав доступу ви можете настроїти, які ролі мають ці права у адміністративному розділі Drupal.

Перейдіть до "People" -> "Permissions" у панелі адміністрування Drupal, де ви зможете призначити права доступу вашим ролям користувачів.

За допомогою цієї системи прав доступу, ви зможете контролювати, які користувачі та ролі мають доступ до певних функцій та ресурсів вашого модуля.

Питання для самоконтролю:

1. Що таке модуль в Drupal 8?
2. Що таке роут і як вони працюють?
3. Як використовувати контролер?
4. Що таке коллбек контролера?
5. Що таке форма?
6. Що таке сервіс та як їх використовують?
7. Як задати кастомні права доступу?

3.3. ХУКИ ТА ПОДІЇ

У Drupal 8 хуки є механізмом, який дозволяє модулям впливати на поведінку ядра Drupal та інших модулів. Вони дозволяють модулям виконувати певні дії або змінювати дані під час різних подій в системі.

Основні принципи хуків в Drupal 8:

Іменування хуків:

Хуки в Drupal 8 повинні мати специфічне ім'я. Вони починаються з префіксу `hook_`, за яким слідує назва хука. Наприклад, хук `hook_node_insert` використовується для виконання дій після вставки нового запису в базу даних для вузла.

Оголошення хуків:

Для оголошення хуків в вашому модулі створіть файл `my_module.module` у корені модуля.

В цьому файлі ви можете визначити функції з іменами, які відповідають іменам хуків.

Наприклад, для оголошення хука `hook_node_insert` ви можете створити функцію `my_module_node_insert()`.

Використання хуків:

Хук може бути викликаний системою або іншим модулем в певному контексті або під час певної події.

Ви можете використовувати хук для виконання певних дій, зміни даних, додавання або модифікації функціональності.

Під час виклику хука, система передає параметри функції, що відповідає хуку, які містять контекстну інформацію та дані, які можуть бути змінені або використані.

Вплив на поведінку:

Хуки дозволяють модулям впливати на поведінку інших модулів або ядра Drupal.

Ви можете змінювати значення певних змінних, додавати або змінювати функціональність, вносити зміни до рендерингу сторінок тощо.

Хуки є потужним інструментом розширення функціональності Drupal 8 і дозволяють вам легко впливати на роботу системи та інших модулів. Існує велика кількість вбудованих хуків у ядрі Drupal та розширюваних модулях, і ви також можете створювати свої власні хуки для забезпечення необхідної функціональності.

У Drupal 8 існує кілька видів хуків, які дозволяють модулям впливати на різні етапи обробки запиту, роботу з контентом, рендеринг сторінок та інші аспекти функціонування системи. Ось деякі з найпоширеніших видів хуків в Drupal 8:

Хуки подій (Event hooks):

Ці хуки викликаються під час виникнення певної події або дії в системі.

Наприклад, `hook_node_insert` викликається після вставки нового вузла в базу даних.

Інші приклади: `hook_form_alter` (для зміни форми), `hook_user_login` (при успішному вході в систему користувача).

Хуки теми (Theme hooks):

- Ці хуки використовуються для зміни або розширення рендерингу сторінок і їх елементів.
- Наприклад, `hook_preprocess_page` дозволяє модифікувати зміст сторінки перед її відображенням.
- Інші приклади: `hook_preprocess_node` (для обробки вузлів), `hook_preprocess_block` (для обробки блоків).

Хуки форм (Form hooks):

- Ці хуки дозволяють модифікувати форми перед їх відображенням або обробкою даних, які були надіслані формою.
- Наприклад, `hook_form_alter` дозволяє змінювати атрибути та поведінку форми.
- Інші приклади: `hook_form_FORM_ID_alter` (для конкретної форми), `hook_form_FORM_ID_validate` (для валідації форми).

Хуки маршрутів (Route hooks):

- Ці хуки використовуються для додавання та зміни маршрутів в системі.
- Наприклад, `hook_menu_local_tasks_alter` дозволяє змінювати вкладки в адміністративному інтерфейсі.
- Інший приклад: `hook_menu_links_discovered_alter` (для зміни посилань у меню).

Хуки бази даних (Database hooks):

- Ці хуки використовуються для модифікації або взаємодії з базою даних.
- Наприклад, `hook_schema` визначає структуру таблиць бази даних для модуля.
- Інший приклад: `hook_entity_update` (при оновленні сутності).

Це лише деякі з видів хуків, які доступні в Drupal 8. Існує багато інших хуків, які дозволяють вам впливати на різні етапи обробки запитів, генерацію вмісту та взаємодію з іншими модулями. Кожен хук має свої унікальні особливості та контекстну інформацію, яку можна використовувати для досягнення потрібної функціональності.

У Drupal 8 використовується механізм подій та прослуховувачів (event listeners) для обробки подій і взаємодії з ними. Цей механізм базується на Symfony EventDispatcher Component і дозволяє модулям реагувати на певні події, які виникають у системі. Основні поняття, які пов'язані з подіями та прослуховувачами в Drupal 8:

Події (Events):

- Події є ключовими моментами або діями, які відбуваються в системі Drupal.
- Приклади подій в Drupal 8: `kernel.request` (коли отримано HTTP-запит), `node.insert` (коли вузол вставлено в базу даних).
- Drupal 8 також надає свої власні вбудовані події, а також можливість створення власних.

Процесори подій (Event Dispatchers):

- Процесори подій використовуються для розсилки (dispatching) подій у системі.
- Drupal 8 використовує Symfony EventDispatcher Component для цього.
- Процесор подій бере на себе відповідність за визначення, коли і які події мають бути відправлені.

Прослуховувачі подій (Event Listeners):

- Прослуховувачі подій це класи, які мають методи для обробки певних подій.
- Прослуховувачі реєструються для слухання певних подій і виконують певну логіку при їх виникненні.
- У Drupal 8, прослуховувачі подій виконують роль "хуків" для обробки певних подій.

Реєстрація прослуховувачів подій:

- Реєстрація прослуховувачів подій в Drupal 8 відбувається за допомогою сервісів.
- Ви можете створити свій власний прослуховувач подій, який реалізує інтерфейс `EventSubscriberInterface`.

- Ваш прослуховувач повинен бути зареєстрований в сервісному контейнері, щоб бути активним.

Обробка подій:

- Коли подія відправлена, прослуховувачі, зареєстровані для цієї події, обробляють її.
- Кожен прослуховувач має методи, які виконуються при виникненні певної події.
- Процесор подій викликає ці методи прослуховувачів, передаючи відповідні дані про подію.

За допомогою механізму подій та прослуховувачів в Drupal 8, ви можете створювати модулі, які ефективно реагують на події в системі та виконують необхідні дії для забезпечення потрібного функціоналу.

Хуки та прослуховувачі подій (event listeners) є двома різними механізмами в Drupal 8 для реагування на події та впливу на поведінку системи. Ось порівняння між хуками та прослуховувачами подій:

Хуки (Hooks):

- Система: Хуки використовуються як механізм розширення функціональності, вбудований у ядро Drupal.
- Оголошення: Хуки оголошуються в модульному файлі `.module` модуля, їхні назви мають специфічний формат, починаючись з `hook_`.
- Взаємодія: Хуки викликаються системою Drupal або іншими модулями в певних контекстах або подіях.
- Зміна даних: Хуки можуть змінювати дані, модифікувати вміст, додавати або змінювати функціональність під час виклику.
- Розподіленість: Хуки розподілені по всій системі Drupal, і інші модулі можуть відгукуватись на них.

Прослуховувачі подій (Event Listeners):

- Система: Прослуховувачі подій базуються на Symfony EventDispatcher Component і використовуються для роботи з подіями в Drupal 8.
- Оголошення: Прослуховувачі подій реалізують інтерфейс `EventSubscriberInterface` і реєструються в сервісному контейнері.
- Взаємодія: Прослуховувачі активуються при виникненні конкретних подій і обробляють їх.

- Зміна даних: Прослуховувачі можуть впливати на дані подій, змінювати їх або виконувати певну логіку обробки.
- Ізольованість: Прослуховувачі подій мають більш ізольовану структуру, що полегшує тестування та керування.

Хоча хуки та прослуховувачі подій виконують схожу функцію — вплив на поведінку системи, вони мають різні підходи до реалізації та взаємодії з подіями. Вибір між хуками та прослуховувачами залежить від контексту і вимог вашого проекту.

Правильне використання хуків в Drupal 8 вимагає врахування кількох підходів та дотримання деяких рекомендацій. Ось кілька правильних підходів:

Іменування хуків:

- Хуки в Drupal 8 мають специфічні назви, які починаються з префіксу `hook_`. Назва хука повинна ясно відображати його функціональність та мету.
- Наприклад, якщо ви створюєте хук для зміни форми, назва хука може бути `hook_form_alter`.

Використання модульної структури:

- Рекомендується розміщувати код хуків відповідного модуля. Створіть `.module` файл у корені модуля для оголошення та виконання хуків.
- Це допоможе зберігати код модуля організованим та підтримуваним.

Документування хуків:

- Для кожного хука ви маєте докладно описати його функціональність та параметри в коментарях коду.
- Це полегшить іншим розробникам зрозуміти призначення хука та правильно використовувати його.

Обробка параметрів:

- Уважно розгляньте параметри, які передаються до функції хука. Використовуйте їх для отримання необхідних даних та контексту.
- Не змінюйте параметри, які передаються по ссилці, якщо це необхідно для вашого випадку використання.

Необхідність виклику хуків:

- Уникайте зайвих викликів хуків, які можуть призвести до непотрібного збільшення навантаження на систему.
- Викликайте хуки тільки тоді, коли це дійсно потрібно для виконання певного завдання.

Розумна обробка:

- Використовуйте хуки для розширення функціональності або модифікації даних у межах вашого модуля.
- Для реалізації бізнес-логіки рекомендується використовувати сервіси та інші компоненти Drupal, а хуки використовувати для зовнішнього впливу на систему.

Ці підходи сприяють чистоті коду, його розширюваності та підтримуваності. Ретельне дотримання цих рекомендацій допоможе вам створити якісні модулі в Drupal.

Питання для самоконтролю:

1. Що таке хук в Drupal 8?
2. Які бувають хуки?
3. Що таке event listeners та dispatchers?
4. Яка різниця між хуками та event listeners?

3.4. РОБОТА З СУТНОСТЯМИ. ENTITY API

У Drupal 8 сутності є ключовою складовою для управління та організації структурованих даних. Вони представляють конкретні типи даних, такі як вузли, користувачі, терміни таксономії тощо. Ось кілька важливих аспектів, пов'язаних з сутностями в Drupal 8:

Властивості сутностей:

- Сутності мають властивості, які описують їх характеристики та дані.
- Наприклад, сутність "нода" має властивості, такі як заголовок, вміст, автор та дата створення.

Сутність як клас:

- Кожен тип сутності має свій клас, який реалізує відповідний інтерфейс.
- Клас сутності містить методи для отримання та зміни властивостей, а також для виконання дій, пов'язаних зі сутністю.

Базові сутності:

- Drupal 8 надає базові сутності, такі як "вузол" (`Node`), "користувач" (`User`), "таксономічний термін" (`Term`) тощо.
- Ці базові сутності вже мають визначені властивості та функціональність, що полегшує роботу з ними.

Визначення власних сутностей:

- Ви також можете створювати власні типи сутностей з визначеними властивостями та поведінкою.
- Визначення власних сутностей включає оголошення сутності у YAML-файлі та створення відповідного класу сутності.

Маніпуляція з сутностями:

- Для маніпуляції з сутностями використовуються методи і сервіси, які надаються Drupal.
- Ви можете здійснювати операції збереження, вилучення, оновлення та вибірки сутностей за допомогою сервісів, таких як `entity_type.manager`.

Форми сутностей:

- Drupal 8 автоматично створює форми для редагування сутностей, які можна використовувати для взаємодії з даними сутності.
- Форми сутностей автоматично забезпечують перевірку даних та збереження змін.

Сутності є потужним механізмом в Drupal 8, який дозволяє організувати та управляти структурованими даними. Вони дозволяють розширювати та модифікувати функціональність Drupal за допомогою власних типів сутностей.

Entity API в Drupal 8 є набором інструментів, які дозволяють працювати з сутностями та виконувати різні операції з ними. Вона надає зручний спосіб створення, збереження, оновлення та вилучення сутностей в Drupal 8. Ось кілька ключових аспектів Entity API:

Менеджер сутностей (Entity Manager):

- Менеджер сутностей (`entity_type.manager`) є сервісом, який надає доступ до різних функцій Entity API.
- Він дозволяє отримувати доступ до визначених типів сутностей, створювати нові сутності, здійснювати пошук і фільтрацію, виконувати операції збереження, оновлення та видалення.

Завантаження сутності (Entity Load):

- За допомогою Entity API ви можете завантажувати сутності за їхнім ідентифікатором або за допомогою певних умов пошуку.
- Ви можете використовувати методи, такі як `load()` або `loadMultiple()`, для отримання сутності з бази даних.

Збереження та оновлення сутності (Entity Save/Update):

- Entity API дозволяє зберігати нові сутності або оновлювати існуючі в базі даних.
- Ви можете використовувати метод `save()` для збереження нової сутності або `save()` для оновлення існуючої.

Видалення сутності (Entity Delete):

- Entity API дозволяє видаляти сутності з бази даних за допомогою методу `delete()`.
- Видалення сутності може бути поодиноким або виконуватися для декількох сутностей одночасно.

Валідація сутності (Entity Validation):

- Entity API надає механізм валідації сутностей перед збереженням.
- Ви можете використовувати функцію `validate()` для перевірки правильності даних сутності перед збереженням.

Події сутностей (Entity Events):

- Entity API надає систему подій, які дозволяють реагувати на дії з сутностями, такі як збереження, оновлення або видалення.
- Ви можете використовувати хуки або прослуховувачі подій для обробки цих дій з сутностями.

Entity API є потужним інструментом в Drupal 8, який спрощує роботу з сутностями та дозволяє здійснювати різноманітні операції з ними. Використання Entity API допомагає розробникам швидко та ефективно працювати з даними сутностей у Drupal 8.

Розглянемо приклад коду, який демонструє завантаження та редагування вузла за допомогою Entity Type Manager в Drupal 8.

У цьому прикладі ми використовуємо сервіс `entity_type.manager`, який передається в функцію як залежність через Dependency Injection. За допомогою цього сервісу ми отримуємо доступ до Entity Type Manager. За допомогою `getStorage()` ми отримуємо сховище для типу сутності "node". Потім ми завантажуємо вузол за допомогою методу `load()`, передаючи його ідентифікатор.

Після завантаження вузла ми можемо змінювати його властивості, встановлюючи новий заголовок та опис. Нарешті, ми викликаємо метод `save()` для збереження змін вузла.

У цьому прикладі також використовується сервіс `messenger`, щоб вивести повідомлення про результат дії. Зверніть увагу, що цей код потребує налаштування

Dependency Injection для отримання сервісу `entity_type.manager` відповідно до вашого модулю.

```
use Drupal\Core\Entity\EntityTypeManagerInterface;

/**
 * @param \Drupal\Core\Entity\EntityTypeManagerInterface $entityTypeManager
 */
function example_function(EntityTypeManagerInterface $entityTypeManager) {
  // Завантаження вузла за його ідентифікатором (наприклад, 123).
  $node = $entityTypeManager->getStorage('node')->load(123);

  // Перевірка, чи вузол було успішно завантажено.
  if ($node instanceof \Drupal\node\NodeInterface) {
    // Зміна властивостей вузла.
    $node->setTitle('Новий заголовок');
    $node->set('field_description', 'Новий опис');

    // Збереження змін вузла.
    $node->save();

    // Виведення повідомлення про успішне збереження.
    \Drupal::messenger()->addMessage('Вузол успішно оновлено.');
  }
  else {
    \Drupal::messenger()->addError('Помилка: Вузол не знайдено.');
  }
}
```

Для оголошення власного типу сутності на основі наслідування класу EntityBase в Drupal 8 потрібно виконати кілька кроків:

Створіть власний модуль:

Створіть папку вашого модуля у директорії `modules/custom` вашого Drupal-сайту.

Додайте файли `your_module.info.yml` та `your_module.module` для оголошення модуля.

Оголосіть власний тип сутності:

Створіть файл `your_module/src/Entity/YourEntity.php` і оголосіть клас вашої сутності, який наслідується від `Drupal\Core\Entity\EntityBase`.

У цьому класі ви можете визначити властивості, методи доступу до них та додаткову функціональність вашої сутності.

```
namespace Drupal\your_module\Entity;
use Drupal\Core\Entity\EntityBase;

/**
 * Defines the YourEntity entity.
 *
 * @ingroup your_module
 *
 * @ContentEntityType(
 *   id = "your_entity",
 *   label = @Translation("Your Entity"),
 *   base_table = "your_entity",
 *   entity_keys = {
 *     "id" = "id",
 *     "label" = "name",
 *   },
 * )
 */
class YourEntity extends EntityBase {
  // Оголошення властивостей та методів вашої сутності
}
```

Оголосіть сховище для вашої сутності:

Створіть файл `your_module/src/YourEntityStorage.php` і оголосіть клас вашого сховища сутності, який наслідується від `Drupal\Core\Entity\EntityStorageBase`.

У цьому класі ви можете визначити додаткові методи для роботи зі сховищем вашої сутності.

```
namespace Drupal\your_module;
use Drupal\Core\Entity\EntityStorageBase;

/**
 * Defines the storage handler class for YourEntity entities.
 */
class YourEntityStorage extends EntityStorageBase {
  // Оголошення методів сховища вашої сутності
}
```

Зареєструйте вашу сутність та сховище:

У файлі `your_module/your_module.module` додайте наступний код для реєстрації вашої сутності та сховища.

У цьому прикладі ми використовуємо хуки `hook_entity_type_build` та `hook_entity_type_alter` для реєстрації вашої сутності та сховища. У функції `your_module_entity_type_build` ми використовуємо метод `registerEntityType()` для оголошення вашої сутності. У функції `your_module_entity_type_alter` ми встановлюємо ваше сховище для сутності за допомогою `setHandlerClass()`.

Це базовий приклад оголошення власного типу сутності на основі наслідування класу EntityBase в Drupal 8. Залежно від вашого конкретного випадку використання та потреб, можуть бути потрібні додаткові налаштування та розширення вашої сутності.

```
use Drupal\Core\Entity\EntityTypeInterface;
use Drupal\Core\Entity\EntityTypeManagerInterface;
use Drupal\Core\Routing\RouteProviderInterface;
use Symfony\Component\DependencyInjection\ContainerInterface;
/**
 * Implements hook_entity_type_alter().
 */
function your_module_entity_type_alter(array &$entity_types) {
  // Додаткові зміни для вашої сутності
```

```
    $entity_types['your_entity']->setHandlerClass('storage',
'Drupal\your_module\YourEntityStorage');
    }
    /**
     * Implements hook_entity_type_build().
     */
    function your_module_entity_type_build(EntityTypeInterface $entity_type,
EntityTypeManagerInterface $entity_type_manager, RouteProviderInterface
$route_provider) {
      if (!$entity_type_manager->hasDefinition('your_entity')) {
        $entity_type_manager->registerEntityType(
          'your_entity',
          'YourEntity',
          [
            'entity_keys' => [
              'id' => 'id',
              'label' => 'name',
            ],
            'base_table' => 'your_entity',
          ]
        );
      }
    }
```

При роботі з сутностями в Drupal 8 варто звертати увагу на кілька важливих моментів, що допоможуть забезпечити правильну та ефективну роботу з сутностями. Ось кілька важливих моментів:

Кешування:

- Drupal 8 використовує систему кешування для покращення продуктивності.
- Пам'ятайте, що при зміні даних сутності ви маєте оновлювати кеш відповідних компонентів, щоб забезпечити відображення оновлених даних.

Права доступу:

- Переконайтеся, що належні права доступу налаштовані для вашої сутності.
- Використовуйте систему ролей та дозволів Drupal для контролю доступу до сутностей та їх даних.

Валідація:

- Забезпечте правильну валідацію даних, щоб гарантувати їхню цілісність та правильність перед збереженням.
- Використовуйте вбудовані механізми валідації Drupal або власні правила валідації для перевірки даних сутностей.

Розширення:

- Drupal 8 надає можливості для розширення функціональності сутностей за допомогою хуків або подій сутностей.
- Використовуйте ці механізми для модифікації або додавання додаткової логіки до операцій з сутностями.

Використання Entity API:

- Використовуйте Entity API та його класи та методи для зручного доступу до функціональності сутностей.
- Наприклад, використовуйте Entity Type Manager для завантаження та збереження сутностей, або Entity Field Manager для роботи з полями сутностей.

Оптимізація запитів:

- При роботі з великою кількістю сутностей, зверніть увагу на оптимізацію запитів до бази даних.
- Використовуйте методи ефективного завантаження та вибірки сутностей, такі як `loadMultiple()` або `loadByProperties()`, щоб уникнути зайвих запитів до бази даних.

Ці важливі моменти допоможуть вам раціонально та ефективно працювати з сутностями в Drupal 8 та забезпечити правильну та надійну роботу з даними.

Питання для самоконтролю:

1. Як завантажити сутність в Drupal 8?
2. Як працювати із сутностями?
3. Як оголосити власну кастомну сутність?

3.5. РОБОТА З БАЗОЮ ДАНИХ. DATABASE API

Робота з базою даних в Drupal 8 відбувається через використання різних компонентів і інструментів, які надаються фреймворком. Ось декілька ключових моментів, пов'язаних з роботою з базою даних в Drupal 8:

Конфігурація бази даних:

- Конфігурація бази даних відбувається у файлі `settings.php`, який знаходиться в директорії `sites/default` вашого Drupal-сайту.
- У цьому файлі ви можете вказати параметри підключення до бази даних, такі як тип бази даних, хост, користувач, пароль тощо.

Використання сервісу Database:

- Drupal 8 надає сервіс `database`, який дозволяє взаємодіяти з базою даних.
- Ви можете використовувати сервіс `database` для виконання запитів до бази даних, вставки, оновлення та вилучення даних.

```
use Drupal\Core\Database\Connection;

/**
 * @param \Drupal\Core\Database\Connection $database
 */
function example_function(Connection $database) {
  // Приклад виконання запиту SELECT до бази даних.
  $query = $database->select('my_table', 't')
    ->fields('t', ['id', 'name'])
    ->condition('status', 1)
    ->execute()
    ->fetchAll();

  foreach ($query as $row) {
    // Робота з результатами запиту.
    $id = $row->id;
    $name = $row->name;
  }
```

```
  // Приклад виконання запиту INSERT до бази даних.
  $database->insert('my_table')
    ->fields(['name' => 'John', 'email' => 'john@example.com'])
    ->execute();

  // Приклад виконання запиту UPDATE до бази даних.
  $database->update('my_table')
    ->fields(['name' => 'John Doe'])
    ->condition('id', 123)
    ->execute();

  // Приклад виконання запиту DELETE до бази даних.
  $database->delete('my_table')
    ->condition('id', 123)
    ->execute();
}
```

Використання сховища сутностей:

- Drupal 8 надає сховища сутностей, які спрощують роботу з даними сутностей в базі даних.
- Ви можете використовувати сховища сутностей для завантаження, збереження, оновлення та вилучення сутностей без прямої роботи з запитами до бази даних.

Міграції бази даних:

- Drupal 8 надає модуль Migrate, який дозволяє здійснювати міграцію даних з інших джерел до Drupal-сайту.
- Ви можете використовувати Migrate для імпорту даних з CSV-файлів, SQL-запитів, веб-служб та інших джерел.

Робота з базою даних в Drupal 8 включає виконання запитів до бази даних, використання сховищ сутностей, конфігурацію бази даних у файлі `settings.php` та можливість міграції даних з використанням модуля Migrate. Зверніть увагу, що в приведеному прикладі використовується базовий синтаксис і наведений код може бути модифікований відповідно до вашого конкретного випадку.

В Drupal 8 ви можете побудувати складні запити до бази даних з використанням методу `join()` об'єкта запиту (Query). Ви можете використовувати метод `join()` для об'єднання таблиць та виконання складних умов.

Ось приклад, як ви можете побудувати запит з використанням join:

```
use Drupal\Core\Database\Database;
use Drupal\Core\Database\Query\SelectInterface;

// Створіть об'єкт запиту.
$query = Database::getConnection()->select('node', 'n');
// Додайте поля, які ви хочете вибрати.
$query->fields('n', ['nid', 'title']);
// Додайте об'єднання з іншими таблицями за допомогою join.
$query->join('field_data_field_example', 'f', 'n.nid = f.entity_id');
$query->join('taxonomy_term_data', 't', 'f.field_example_target_id = t.tid');

// Вкажіть поля, які ви хочете вибрати з об'єднаних таблиць.
$query->fields('f', ['field_example_value']);
$query->fields('t', ['name']);

// Додайте умови до запиту.
$query->condition('n.type', 'article');
$query->condition('t.vid', 1);
// Виконайте запит.
$result = $query->execute();

// Отримайте результати запиту.
foreach ($result as $record) {
  // Обробка результатів запиту.
  $nid = $record->nid;
  $title = $record->title;
  $fieldValue = $record->field_example_value;
  $termName = $record->name;
}
```

У цьому прикладі ми створюємо об'єкт запиту, вказуємо таблицю `"node"` і позначаємо її як `"n"` для короткого використання. Далі ми додаємо поля, які ми хочемо вибрати з таблиці `"node"`. Потім ми використовуємо метод `join()`, щоб об'єднати таблиці `"field_data_field_example"` та `"taxonomy_term_data"` з таблицею `"node"` за допомогою відповідних умов з'єднання. Ми також вказуємо поля, які ми хочемо вибрати з об'єднаних таблиць.

Далі, ми додаємо умови до запиту за допомогою методу `condition()`. Нарешті, ми виконуємо запит за допомогою методу `execute()`. Результати запиту можна обробляти в циклі `foreach`.

Використання методу `join()` дозволяє вам побудувати складні запити до бази даних, об'єднуючи таблиці та встановлюючи відповідні умови з'єднання. Зверніть увагу, що ви можете використовувати різні методи запиту (наприклад, `condition()`, `orderBy()`, `groupBy()`), щоб додати додаткові умови та налаштування до вашого запиту.

В Drupal 8 ви можете використовувати як статичні, так і динамічні запити до баз даних для взаємодії з даними. Ось розбіжності між цими двома підходами:

Статичні запити:

- Статичні запити будуються за допомогою класу `\Drupal\Core\Database\Connection` та методу `query()`.
- Запити написані у вигляді рядків SQL-коду.
- Потребують ручного виконання екранування параметрів для запобігання SQL-ін'єкціям.

Ось приклад статичного запиту:

```
use Drupal\Core\Database\Database;

// Отримання об'єкту підключення до бази даних.
$database = Database::getConnection();
// Побудова статичного запиту.
$query = $database->query('SELECT * FROM {my_table} WHERE id = :id', [':id' => 123]);

// Виконання запиту та отримання результатів.
$results = $query->fetchAll();
```

У цьому прикладі ми використовуємо метод `query()` для побудови статичного SQL-запиту до бази даних. Ми вказуємо таблицю `{my_table}` та параметр `:id`, який буде підставлений за допомогою масиву параметрів.

Динамічні запити:

- Динамічні запити будуються за допомогою класу `\Drupal\Core\Database\Query\Select` або інших класів запиту.
- Запити побудовані з використанням методів класу запиту.
- Параметри автоматично екрануються для запобігання SQL-ін'єкціям.

Ось приклад динамічного запиту:

```
use Drupal\Core\Database\Database;
use Drupal\Core\Database\Query\Condition;
use Drupal\Core\Database\Query\SelectInterface;

// Отримання об'єкту підключення до бази даних.
$database = Database::getConnection();

// Побудова динамічного запиту.
$query = $database->select('my_table', 't')
  ->fields('t', ['id', 'name'])
  ->condition('id', 123)
  ->condition('status', 1, '=');

// Виконання запиту та отримання результатів.
$results = $query->execute()->fetchAll();
```

У цьому прикладі ми використовуємо клас `\Drupal\Core\Database\Query\Select` для побудови динамічного запиту до бази даних

Ми вказуємо таблицю `'my_table'`, вибираємо поля `'id'` та `'name'`, а також додаємо умови за допомогою методу `condition()`.

В обох випадках статичних і динамічних запитів важливо пам'ятати про безпеку і захист від SQL-ін'єкцій. Використовуйте екранування параметрів або параметризовані запити для запобігання небезпечним запитам.

Вибір між статичними і динамічними запитами до бази даних в Drupal 8 залежить від вашої конкретної ситуації та вимог вашого проекту. Ось деякі розгляди, які можуть вам допомогти визначити, коли використовувати кожен з цих підходів:

Статичні запити (SQL-рядки):

- Використовуйте статичні запити, коли вам потрібно виконати простий запит до бази даних з фіксованим SQL-кодом.
- Цей підхід може бути зручним, коли ви маєте заздалегідь відомі запити, які не потребують складної динаміки або генерації запиту з допомогою умов та змінних.

Динамічні запити (об'єкти запиту):

- Використовуйте динамічні запити, коли вам потрібно генерувати складні запити до бази даних залежно від умов, фільтрів, сортування та інших динамічних факторів.
- Цей підхід є більш гнучким і потужним, оскільки ви можете динамічно змінювати та налаштовувати ваш запит в залежності від умов та змінних в вашому коді.

Вибір між статичними і динамічними запитами залежить від ваших потреб, складності запиту та рівня гнучкості, який вам потрібно. Узгоджуйте використання статичних або динамічних запитів з вимогами вашого проекту та враховуйте плюси та мінуси кожного підходу.

Питання для самоконтролю:

1. Як виконати запит до бази даних в Drupal 8?
2. Що таке статичні та динамічні запити?
3. Як будувати складні запити із Join?

3.6. РОБОТА З CONFIGURATION API

Configuration API в Drupal 8 надає зручний спосіб управління конфігурацією вашого сайту. Він дозволяє зберігати та зчитувати конфігураційні дані у структурованому форматі. Ось детальніша інформація про Configuration API:

Конфігураційні об'єкти:

- Конфігураційні дані зберігаються у вигляді об'єктів `ConfigEntityInterface` або `ConfigEntityBase`.
- Кожен конфігураційний об'єкт представляє окрему конфігураційну сутність, таку як налаштування модуля або теми.

Файли конфігурації:

- Файли конфігурації зберігаються у директорії `config/install` або `config/optional` вашого модуля або теми.
- Кожен файл конфігурації містить серіалізовані дані конфігурації у форматі YAML.

Маніпулювання конфігурацією:

- Для зчитування та зміни конфігураційних даних використовуються методи з класу `ConfigFactory`.
- Наприклад, ви можете зчитати конфігурацію модуля за допомогою `$configFactory->get('module.name')` та змінити значення конфігурації за допомогою `$config->set('key', 'value')`.

Структуровані дані конфігурації:

- Configuration API дозволяє визначати схеми конфігурації за допомогою YAML-файлів.
- Схеми дозволяють встановлювати типи даних, обов'язкові поля, значення за замовчуванням та інші правила валідації.

Експорт та імпорт конфігурації:

- Конфігурацію можна експортувати у вигляді YAML-файлів та імпортувати з них.
- Це дозволяє керувати конфігурацією у версійному контролі, робити резервні копії та обмінюватися конфігурацією між різними середовищами.

Захист конфігурації:

- Для забезпечення безпеки конфігурації використовується механізм прав доступу.
- Конфігурація може бути доступна лише певним ролям або користувачам.

Configuration API в Drupal 8 дозволяє зручно управляти конфігураційними даними вашого сайту, зберігати їх у структурованому форматі, експортувати та імпортувати. Він допомагає забезпечити стійкість та портабельність вашої конфігурації між різними середовищами розробки.

Розглянемо приклад використання конфігурацій в коді Drupal 8:

```
use Drupal\Core\Config\ConfigFactoryInterface;

/**
 * Example function that uses configuration.
 */
function example_function(ConfigFactoryInterface $configFactory) {
  // Зчитування конфігурації модуля.
  $config = $configFactory->get('module.name');

  // Отримання значення конфігураційного параметра.
  $paramValue = $config->get('param_name');

  // Зміна значення конфігураційного параметра.
  $config->set('param_name', 'new_value');
  $config->save();
}
```

У цьому прикладі ми використовуємо `ConfigFactoryInterface`, який надає доступ до сервісу конфігурації (`ConfigFactory`). У функції `example_function()` ми використовуємо `$configFactory` для отримання конфігураційного об'єкту за допомогою `get()`, де `''module.name''` - це ідентифікатор конфігурації модуля.

Потім ми можемо використовувати метод `get()` на конфігураційному об'єкті для отримання значення певного параметра конфігурації (`'param_name''). Метод `set()` дозволяє змінювати значення параметра, і після цього ми викликаємо `save()`, щоб зберегти зміни.

Використання конфігурацій в Drupal 8 має свої переваги і недоліки, які варто враховувати при розробці проекту.

Переваги використання конфігурацій:

Збереження налаштувань: Конфігураційна система дозволяє зберігати налаштування сайту у структурованому форматі. Це дозволяє легко змінювати і зберігати конфігурацію у версійному контролі.

Портабельність: Конфігураційні дані можна експортувати та імпортувати між різними середовищами розробки. Це дозволяє легко переносити конфігурацію з одного сайту на інший.

Захист даних: Конфігураційні дані можуть бути обмежені доступом тільки для визначених ролей або користувачів. Це допомагає забезпечити безпеку і захист конфіденційної інформації.

Гнучкість: Конфігураційна система дозволяє змінювати налаштування сайту без необхідності внесення змін у код. Це робить ваш сайт більш гнучким і дозволяє адміністраторам змінювати налаштування без втручання розробників.

Недоліки використання конфігурацій:

Складність міграції: Перенесення конфігурації між різними версіями Drupal може бути викликом. Іноді необхідні додаткові кроки для забезпечення сумісності та правильної міграції конфігураційних даних.

Взаємозалежності: У разі, коли конфігураційні дані взаємозалежні, зміни в одній частині конфігурації можуть мати вплив на інші частини. Це може вимагати уважного управління та тестування змін в конфігурації.

Обмеження формату: Конфігураційна система Drupal використовує формат YAML для збереження конфігураційних даних. Це означає, що дані повинні відповідати синтаксису YAML і дотримуватися його обмежень.

Вплив на продуктивність: Збереження та зчитування конфігураційних даних може мати певний вплив на продуктивність, особливо при роботі з великою кількістю конфігураційних об'єктів. Продумане кешування та оптимізація можуть бути важливими для забезпечення ефективності.

Питання для самоконтролю:

1. Що таке конфігурації в Drupal 8?
2. Як працювати з конфігураціями в коді?
3. Які переваги і недоліки використання конфігурацій?

3.7. РОБОТА З STATE API

State API в Drupal 8 надає зручний спосіб зберігання та отримання стану вашого сайту. Він дозволяє зберігати дані на довший термін, ніж сесії, але коротший, ніж конфігурація.

Зберігання стану:

- State API дозволяє зберігати дані, які пов'язані зі станом вашого сайту, наприклад, налаштування модуля, прогрес виконання завдання або інші тимчасові дані.
- Дані зберігаються у базі даних, а не в сесіях, тому вони доступні на всіх сторінках сайту.

Структуровані дані:

- Дані, які зберігаються за допомогою State API, можуть бути структуровані за допомогою масивів або об'єктів.
- Це дозволяє вам зберігати складні дані та ієрархічну структуру.

Доступ до даних:

- Доступ до даних здійснюється за допомогою сервісу `state`.
- Ви можете отримати значення стану за допомогою методу `get('key')` та встановити значення за допомогою методу `set('key', 'value')`.

Скопіювання до кешу:

- Дані стану можуть бути копійовані до кешу, щоб покращити продуктивність.
- Ви можете використовувати метод `setMultiple()` для копіювання багатьох значень одночасно.

Життєвий цикл даних:

- Дані стану не зберігаються назавжди. Вони можуть бути видалені з бази даних після певного періоду часу або при спеціальних умовах.
- Використовуйте State API для збереження тимчасових або необхідних налаштувань, а не для довгострокового зберігання даних.

State API в Drupal 8 дозволяє зберігати тимчасові дані та налаштування, пов'язані зі станом вашого сайту. Використовуйте його для збереження даних, які не потребують тривалого збереження, але важливі для функціонування вашого сайту.

State API та Configuration API в Drupal 8 мають різні призначення та використовуються для різних цілей. Ось розбіжності між ними:

Configuration API:

- Використовується для зберігання налаштувань сайту та модулів.
- Конфігураційні дані зберігаються у структурованому форматі і доступні на всіх сторінках сайту.
- Можна експортувати та імпортувати конфігурацію між різними середовищами розробки.
- Дані конфігурації зазвичай змінюються розробниками або адміністраторами сайту.

State API:

- Використовується для зберігання тимчасових даних та стану сайту.
- Дані стану зберігаються у базі даних і можуть бути доступні на всіх сторінках сайту.
- Призначений для збереження даних, які змінюються під час виконання запитів та процесування даних.
- Дані стану зазвичай змінюються під час виконання коду або відповідно до дій користувачів.

Основна різниця між Configuration API та State API полягає в їхньому призначенні. Configuration API використовується для зберігання постійних налаштувань, які не змінюються під час виконання коду або відповідно до дій користувачів. З іншого боку, State API використовується для зберігання тимчасових даних та стану, які можуть змінюватись під час виконання коду або відповідно до дій користувачів.

Вибір між Configuration API та State API залежить від призначення даних та їхньої тривалості. Якщо дані є постійними налаштуваннями, які не змінюються під час виконання коду, варто використовувати Configuration API. Увипадку тимчасових даних та стану, які змінюються під час виконання, State API є кращим варіантом.

Розглянемо приклад використання State API в коді Drupal 8:

У цьому прикладі ми використовуємо State API, передавши об'єкт `StateInterface` до функції `example_function()`.

Метод `set()` дозволяє зберегти значення стану з ключем `'key'` та значенням `'value'`.

Метод `get()` використовується для отримання значення стану за ключем `'key'`.

Метод `has()` перевіряє, чи є значення стану з ключем `''key''`. Виконайте певні дії, якщо значення стану присутнє.

Метод `delete()` використовується для видалення значення стану з ключем `''key''`.

```
use Drupal\Core\State\StateInterface;

/**
 * Example function that uses State API.
 */
function example_function(StateInterface $state) {
  // Збереження даних стану.
  $state->set('key', 'value');

  // Отримання значення стану.
  $value = $state->get('key');

  // Перевірка наявності значення стану.
  if ($state->has('key')) {
    // Виконати певні дії, якщо значення стану присутнє.
  }

  // Видалення значення стану.
  $state->delete('key');
}
```

Питання для самоконтролю:

1. Що таке стани (states) в Drupal 8?
2. Як працювати з станами в коді?
3. В чому різниця між State API та Configuration API?

3.8. РОЗРОБКА ТЕМ

В Drupal 8, тема (theme) є однією з ключових складових системи оформлення і відображення вмісту вашого сайту. Вона визначає зовнішній вигляд та структуру вашого сайту, включаючи макети, стилі, шрифти, колірну схему, зображення та інші елементи дизайну.

Основні характеристики тем в Drupal 8:

Зовнішній вигляд та макети: Тема визначає, як ваш сайт виглядає для користувачів. Вона включає в себе розміщення блоків, шаблони сторінок, розташування елементів і структуру сторінок.

Шаблони: Тема використовує шаблони для відображення різних типів контенту та елементів на сторінці. Шаблони можуть бути написані з використанням HTML, CSS, PHP та спеціальних змінних та функцій Drupal.

Стилі та CSS: Тема включає CSS-файли, які використовуються для стилізації та оформлення вашого сайту. Ви можете встановлювати стилі для різних елементів, класів та ідентифікаторів, щоб досягти бажаного вигляду.

Респонсивний дизайн: Багато тем в Drupal 8 підтримують респонсивний дизайн, що означає, що вони забезпечують оптимальний вигляд та взаємодію на різних пристроях та розмірах екрану.

Конфігурація теми: Ви можете налаштовувати параметри теми через адміністративний інтерфейс Drupal. Це дозволяє вам змінювати кольори, зображення, шрифти, макети та інші аспекти теми без необхідності втручання в код.

Модульність тем: Drupal 8 пропонує можливість успадкування тем, що дозволяє вам створювати теми, які базуються на інших темах. Це спрощує управління темами та дозволяє використовувати властивості та стилі з інших тем.

Розширення теми: Ви можете розширювати функціональність теми, використовуючи тематичні розширення (Theming Extensions) та хуки Drupal. Це дозволяє вам кастомізувати вигляд і поведінку вашої теми.

Теми в Drupal 8 дають вам можливість створювати привабливі та функціональні сайти з вищим рівнем контролю над їхнім виглядом та відображенням.

Початок розробки теми в Drupal 8 можна розпочати з наступних кроків:

Розуміння вимог і дизайну: Перш ніж почати розробку теми, важливо мати чітке розуміння вимог вашого проекту і дизайну, який ви хочете досягти. Вивчіть дизайн-макети, специфікації щодо кольорів, шрифтів, макетів сторінок тощо.

Створення нової теми: Створіть новий каталог для вашої теми в папці `/themes` вашого Drupal-сайту. Дайте темі назву, яка відповідає проекту або її призначенню.

Файл `info.yml`: Створіть файл `yourtheme.info.yml` в каталозі вашої теми. У цьому файлі визначте основні властивості теми, такі як назва, версія, базова тема, регістрація стилів та скриптів.

Шаблони: Створюйте шаблони для різних типів контенту та елементів. Використовуйте розмітку HTML та спеціальні змінні, які надає Drupal, для динамічного відображення змісту.

CSS-стилі: Створюйте CSS-файли для оформлення вашої теми. Використовуйте класи та ідентифікатори, щоб стилізувати елементи, використовуючи CSS-селектори. Враховуйте резпонсивний дизайн, якщо це потрібно.

Файл `page.html.twig`: Створіть шаблон `page.html.twig`, який визначає основну структуру сторінки вашої теми. В цьому шаблоні визначається розміщення блоків, хедер, футер, навігаційне меню тощо.

Налаштування теми: Налаштуйте параметри теми через адміністративний інтерфейс Drupal. Встановіть кольори, шрифти, зображення, макети та інші властивості вашої теми.

Тестування: Перевірте, як ваша тема відображається на різних сторінках та пристроях. Переконайтеся, що всі елементи відображаються правильно, стилі застосовуються належним чином і сайт виглядає так, як очікується.

Вдосконалення теми: Поступово вдосконалюйте вашу тему, додаючи додаткові функціональність, удосконалюючи дизайн та виправляючи помилки.

Документація: Добре задокументуйте вашу тему, включаючи інструкції з використання та налаштування, опис структури файлів, використані стилі та шаблони.

Ці кроки відображають загальну послідовність розробки теми в Drupal 8. Залежно від складності вашого проекту та ваших потреб, ви можете виконувати додаткові кроки або змінювати послідовність робіт.

3.8.1. СТРУКТУРА ТЕМИ

Структура теми в Drupal 8 має свої особливості та конвенції. Ось загальна структура теми в Drupal 8:

Коренева папка теми:

- Назва папки повинна відповідати назві вашої теми.
- Ця папка містить всі файли та підпапки, пов'язані з вашою темою.

Файл `yourtheme.info.yml`:

- Цей файл містить основну інформацію про вашу тему, таку як назва, версія, базова тема, регістрація стилів та скриптів.
- Файл `yourtheme.info.yml` також може містити налаштування кольорів, шрифтів, макетів тощо.

Підпапка `css`:

- Ця папка містить CSS-файли, які використовуються для стилізації вашої теми.
- Рекомендовано структурувати CSS-файли за модульністю, наприклад, створювати окремі файли для стилів окремих компонентів або сторінок.

Підпапка `js`:

- В цій папці можна розмістити JavaScript-файли, які використовуються в вашій темі.
- Рекомендується використовувати бібліотеки, такі як jQuery, як залежності та включати їх у ваші файли JavaScript.

Підпапка `templates`:

- У цій папці зберігаються шаблони (templates) для відображення різних типів контенту та елементів.
- Файли шаблонів повинні мати розширення `.html.twig` та використовувати спеціальні змінні та функції Drupal для відображення змісту.

Підпапка `images`:

- В цій папці можна зберігати зображення, які використовуються в вашій темі.
- Рекомендується підтримувати структуру підпапок для категоризації зображень.

Файл `yourtheme.libraries.yml`:

- Цей файл визначає бібліотеки стилів та скриптів, які використовуються в вашій темі.
- Використовуйте цей файл для реєстрації та підключення залежностей стилів та скриптів.

Файл `yourtheme.breakpoints.yml`:

- Цей файл містить налаштування брейкпоінтів для респонсивного дизайну вашої теми.

- Визначте різні розміри екранів та відповідні CSS-класи для них.

Файл `yourtheme.theme`:

- Цей файл містить функції-хуки та додаткові налаштування теми.
- Ви можете використовувати його для модифікації функціональності та вигляду вашої теми.

В Drupal 8 є кілька популярних базових тем, які можна використовувати як основу для розробки власних тем. Ось декілька прикладів базових тем в Drupal 8:

Classy:

- Classy є базовою темою, яка поставляється з ядром Drupal 8.
- Вона надає базові стилі та шаблони для основних елементів Drupal, таких як форми, таблиці, розмітка сторінки та інші.
- Classy спрощує розробку тем та забезпечує консистентний вигляд з елементами Drupal.

Stable:

- Stable також є базовою темою, яка поставляється з ядром Drupal 8.
- Ця тема забезпечує стабільну базу для розробки тем, зберігаючи сумісність з попередніми версіями Drupal.
- Stable надає мінімальні стилі та шаблони, що дозволяє розробникам повністю контролювати вигляд теми.

Bootstrap:

- Bootstrap є популярною базовою темою, яка інтегрує фреймворк Bootstrap в Drupal.
- Вона надає готові компоненти, стилі та шаблони, що дозволяють швидко розробляти стильні та респонсивні теми.
- Bootstrap дозволяє використовувати готові класи та функціональність Bootstrap для будь-яких елементів вашого сайту.

Zen:

- Zen є іншою популярною базовою темою, яка надає гнучку основу для розробки тем.
- Вона пропонує структуровану систему стилів, розширювані шаблони та інструменти для розробки тем на основі Zen.
- Zen дозволяє вам швидко створювати та налаштовувати теми згідно з вашими потребами.

Це лише кілька прикладів базових тем в Drupal 8. У Drupal.org та інших ресурсах можна знайти ще багато інших базових тем, які можна використовувати у своїх проектах. Вибір базової теми залежить від вашої конкретної потреби, дизайну та вимог вашого проекту.

3.8.2. СТРУКТУРА .INFO.YML

Файл `theme.info.yml` є одним з ключових файлів, що використовуються в Drupal 8 для оголошення теми. В цьому файлі вказуються основні властивості теми, такі як назва, версія, базова тема, регістрація стилів та скриптів. Даний файл дуже схожий до аналогічного за призначенням файлу в структурі модуля.

Основні елементи, які можуть бути включені в `theme.info.yml`:

`name`: Назва теми. Вона повинна бути унікальною і слід враховувати регістр.

`type`: Тип теми, зазвичай встановлюється як `theme`.

`description`: Короткий опис теми, який пояснює її призначення та функціональність.

`core_version_requirement`: Вказує мінімальну версію ядра Drupal, необхідну для використання теми.

`package`: Вказує категорію або пакет, до якого належить тема.

`version`: Версія теми.

`base_theme`: Вказує базову тему, від якої успадковується функціональність та стилі.

`libraries`: Реєструє бібліотеки стилів та скриптів, які використовуються в темі.

`css`: Перелік CSS-файлів, які будуть підключені до сторінок сайту.

`js`: Перелік JavaScript-файлів, які будуть підключені до сторінок сайту.

`regions`: Визначає регіони, які використовуються в темі для розміщення блоків.

`libraries-override`: Дозволяє перевизначати бібліотеки стилів та скриптів, використовувані в базовій темі.

`settings`: Налаштування теми, які можна налаштовувати через адміністративний інтерфейс Drupal.

Файл `theme.info.yml` визначає основні параметри і налаштування теми. Цей файл потрібно розмістити в кореневій папці теми. Після внесення змін до `theme.info.yml`, необхідно провести процес кешування теми для застосування змін.

3.8.3. СТРУКТУРА .THEME

Файл `.theme` в Drupal 8 є одним з ключових файлів теми. Цей файл має формат `yourtheme.theme` (наприклад, `mytheme.theme`) і використовується для розширення функціональності теми та виконання додаткових налаштувань.

Особливості файлу `.theme`:

Розташування: Файл `.theme` повинен бути розміщений в кореневій папці вашої теми.

Виконуваний код: Файл `.theme` містить PHP-код, який виконується під час завантаження теми або під час виконання певних подій або хуків Drupal.

Розширення функціональності: Ви можете використовувати файл `.theme`, щоб додати власні функції, змінити поведінку теми, модифікувати вихідний HTML-код та інше.

Хуки теми: Файл `.theme` дозволяє вам використовувати хуки теми, які дозволяють вам реагувати на певні події або змінювати вихідний код сторінок.

Програмування на рівні теми: Файл `.theme` дає вам можливість програмувати на рівні теми і змінювати функціональність та вигляд вашого сайту без необхідності втручання в код ядра або модулів.

Приклади використання файлу `.theme` в Drupal 8:

- Додавання альтернативних класів до HTML-елементів
- Зміна розміщення блоків
- Модифікація вихідного HTML-коду
- Додавання функціональності з використанням хуків теми
- Налаштування теми з використанням налаштувань, збережених у файлі `.theme`

Файл `.theme` в Drupal 8 дозволяє розширити функціональність теми, вплинути на вигляд та поведінку сайту та забезпечити більшу гнучкість при розробці тем.

За допомогою файлу `.theme` в Drupal 8 ви можете використовувати функції-предпроцесори (preprocess functions) для модифікації даних перед відображенням вихідного HTML-коду сторінок. Ось декілька типових прикладів функцій-предпроцесорів, які можна використовувати в файлі `.theme`:

`yourtheme_preprocess_page(&$variables)`: Ця функція викликається під час предпроцесування шаблону `page.html.twig`. Ви можете використовувати її для модифікації змінних, які передаються у шаблон сторінки, таких як `page`, `node`, `breadcrumb` тощо.

```
function yourtheme_preprocess_page(&$variables) {
  // Додаткові змінні та модифікації даних
  $variables['custom_variable'] = 'Custom Value';
}
```

`yourtheme_preprocess_node(&$variables)`: Ця функція викликається під час предпроцесування шаблону `node.html.twig`. Ви можете використовувати її для модифікації змінних, які передаються у шаблон вузла, таких як `node`, `content`, `title` тощо.

```
function yourtheme_preprocess_node(&$variables) {
  // Додаткові змінні та модифікації даних
  $variables['custom_variable'] = 'Custom Value';
}
```

`yourtheme_preprocess_block(&$variables)`: Ця функція викликається під час предпроцесування шаблону `block.html.twig`. Ви можете використовувати її для модифікації змінних, які передаються у шаблон блоку, таких як `block`, `content` тощо.

```
function yourtheme_preprocess_block(&$variables) {
  // Додаткові змінні та модифікації даних
  $variables['custom_variable'] = 'Custom Value';
}
```

`yourtheme_preprocess_field(&$variables)`: Ця функція викликається під час предпроцесування шаблону `field.html.twig`. Ви можете використовувати її для модифікації змінних, які передаються у шаблон поля, таких як `element`, `items`, `label` тощо.

```
function yourtheme_preprocess_field(&$variables) {
  // Додаткові змінні та модифікації даних
  $variables['custom_variable'] = 'Custom Value';
}
```

3.8.4. ТЕХНОЛОГІЯ TWIG

Twig є шаблонним двигуном, використовуваним в Drupal 8 для генерації вихідного HTML-коду. Він забезпечує розділення логіки відображення та даних в шаблонах, що дозволяє більш ефективно керувати виглядом сторінок та компонентів в Drupal 8.

Основні особливості Twig:

Простий синтаксис: Twig використовує зрозумілий і легко читаємий синтаксис, що дозволяє швидко розробляти та змінювати шаблони.

Розділення логіки відображення: Twig забезпечує розділення логіки відображення (шаблону) та логіки обробки даних (контролера). Це дозволяє вам зосередитися на структурі та вигляді сторінок, відокремивши їх від бізнес-логіки.

Змінні та фільтри: Twig надає багатий набір змінних та фільтрів для маніпуляції даними в шаблонах. Ви можете використовувати змінні для передачі даних з контролера до шаблону та застосовувати фільтри для форматування та обробки даних.

Умови та цикли: Twig підтримує умови (if-else) та цикли (for, foreach), що дозволяють вам контролювати потік виконання та повторювати блоки коду в шаблонах.

Наслідування та включення: Twig дозволяє використовувати наслідування шаблонів, де ви можете створювати базові шаблони зі спільними елементами та розширювати їх у конкретних шаблонах. Також можна використовувати включення, щоб вставляти частини шаблону в інші шаблони.

Функції та хуки: Twig дозволяє використовувати власні функції та хуки в шаблонах. Ви можете створювати власні функції для виконання певних операцій або виклику певних дій. Хуки дозволяють вам модифікувати поведінку шаблону зовнішніми модулями.

Twig є потужним і гнучким шаблонним двигуном, який дозволяє ефективно розробляти та керувати виглядом сторінок і компонентів в Drupal 8.

Синтаксис Twig використовує зрозумілу та просту структуру, що дозволяє легко читати та розробляти шаблони. Основні елементи синтаксису Twig включають наступні:

Виведення даних: Виведення даних здійснюється за допомогою подвійних фігурних дужок `{{ }}`. Наприклад, `{{ variable }}` виведе значення змінної `variable`.

Коментарі: Коментарі в Twig починаються з `{#` і закінчуються на `#}`. Наприклад, `{# This is a comment #}`.

Змінні: Використовуйте імена змінних без префіксу `$`. Наприклад, `{{ username }}` виведе значення змінної `username`.

Умовні оператори: Умовні оператори дозволяють вам виконувати різні дії залежно від певних умов. Використовуйте `{% if condition %}...{% endif %}` для виконання блоку коду, якщо умова вірна.

Цикли: Twig підтримує цикли для повторення блоку коду. Використовуйте `{% for item in array %}...{% endfor %}` для повторення блоку коду для кожного елемента у масиві.

Фільтри: Фільтри дозволяють змінювати або форматувати значення змінних. Використовуйте `{{ variable | filter }}` для застосування фільтру до значення змінної.

Включення: Включення дозволяють вам вставляти вміст іншого шаблону всередину поточного шаблону. Використовуйте `{% include 'template.twig' %}` для включення шаблону з іменем `template.twig`.

Наслідування: Наслідування дозволяють вам створювати базовий шаблон, який можна розширити в інших шаблонах. Використовуйте `{% extends 'base.twig' %}` для успадкування від базового шаблону з іменем `base.twig`.

Це лише кілька прикладів синтаксису Twig. Twig має ще багато інших можливостей, таких як функції, хуки, фільтри та інші конструкції, які дозволяють більш гнучко керувати виглядом і поведінкою шаблонів.

3.8.5. ТЕМПЛЕЙТИ ТА ЇХ ВИКОРИСТАННЯ

У темах Drupal 8 типові темплейти використовуються для відображення різних компонентів сторінок. Ось кілька типових темплейтів, які можна знайти в темі Drupal 8:

1. `html.twig`: Це головний темплейт, який відображає заголовок, метатеги, теги `<head>` і основний вміст сторінки. Він включає в себе шаблони для `<head>`, `<body>` та інших елементів.
2. `page.html.twig`: Цей темплейт використовується для відображення основного контенту сторінки. Він містить звичайну структуру сторінки, яка може містити заголовок, меню, області змісту та інші компоненти.
3. `node.html.twig`: Цей темплейт використовується для відображення вмісту вузлів (наприклад, статей, сторінок). Він може містити заголовок, зміст, зображення та інші поля вузла.

4. `block.html.twig`: Цей темплейт використовується для відображення блоків, які можна розміщувати в різних регіонах сторінки. Він може містити заголовок, зміст та інші елементи блоку.
5. `menu.html.twig`: Цей темплейт використовується для відображення меню. Він містить розмітку та стиль для пунктів меню та їх підрівнів.
6. `field.html.twig`: Цей темплейт використовується для відображення полів контенту, таких як текст, зображення, посилання тощо. Він може містити розмітку для відображення поля та додаткових налаштувань.

У темах Drupal 8 існує певна конвенція іменування (name conventions) для темплейтів, яка дозволяє системі Drupal автоматично розпізнавати та використовувати потрібні темплейти для відображення різних компонентів. Ось деякі загальні правила іменування темплейтів у темах Drupal 8:

Шаблони компонентів: Більшість компонентів мають свій власний шаблон, який можна переопріділити в темі. Шаблони компонентів зазвичай називаються за наступною схемою: `component.html.twig`. Наприклад, `node.html.twig` для вузлів, `block.html.twig` для блоків, `menu.html.twig` для меню тощо.

Шаблони регіонів: Якщо вам потрібно модифікувати відображення конкретного регіону, ви можете створити шаблон для цього регіону. Шаблони регіонів зазвичай називаються за наступною схемою: `region.html.twig`. Наприклад, `header.html.twig` для регіону заголовка, `sidebar.html.twig` для бічного регіону тощо.

Шаблони теми: Іноді може бути потрібно створити спеціальний шаблон для певної теми або деякої сторінки. Шаблони теми зазвичай називаються за наступною схемою: `theme.html.twig`. Наприклад, `page--front.html.twig` для головної сторінки, `page--node.html.twig` для вузлів тощо.

Перевизначення шаблонів: Якщо ви хочете переопріділити стандартний шаблон, використовуйте таку схему іменування: `component--variant.html.twig`. Наприклад, `node--teaser.html.twig` для відображення анонсів вузлів, `block--custom.html.twig` для власного блоку тощо.

Важливо зазначити, що ви можете розмістити ці шаблони в папці `templates` вашої теми.

В Drupal 8, регіони та лайаути використовуються для організації розташування та відображення компонентів на сторінці. Регіони представляють собою контейнери, в які можна розміщувати блоки, форми, меню та інші елементи, а лайаути визначають загальну структуру сторінки.

Основні поняття:

Регіони: Регіони - це місця на сторінці, де ви можете розміщувати компоненти (блоки, форми тощо). Кожна тема може мати свій набір регіонів. Деякі стандартні регіони, які часто використовуються, це `header` (заголовок), `footer` (підвал), `sidebar` (бічний панель) та інші. Регіони визначаються в файлі теми `yourtheme.info.yml`.

Лайаути: Лайаути - це шаблони, які визначають загальну структуру сторінки і включають розміщення регіонів. Лайаути дозволяють вам визначити розміщення та порядок регіонів на сторінці. У Drupal 8 лайаути визначаються в файлі теми `yourtheme.layouts.yml`. Кожен лайаут може мати свою унікальну структуру та розміщення регіонів.

Шаблони регіонів: Для кожного регіону можна створити окремий шаблон, щоб настроїти його вигляд та стиль. Шаблон регіону зазвичай має назву `region.html.twig` і знаходиться в папці `templates` вашої теми. В шаблоні регіону ви можете використовувати змінні, фільтри та цикли для відображення вмісту регіону.

Переопріділення лайаутів та регіонів: Ви можете переопріділяти стандартні лайаути та регіони, створюючи власні варіації. Це дозволяє вам налаштувати розміщення компонентів і структуру сторінки відповідно до потреб вашого проекту.

Робота з регіонами та лайаутами в Drupal 8 дозволяє вам гнучко керувати розміщенням компонентів на сторінці та пристосовувати їх до потреб вашого проекту. Вам потрібно визначити регіони в темі, створити лайаути та налаштувати їх розміщення, а потім використовувати регіони у ваших шаблонах для відображення вмісту.

Правильне проектування темплейтів в темі Drupal 8 є важливим етапом розробки, оскільки воно визначає вигляд та структуру вашого сайту. Ось кілька кроків та рекомендацій, які можуть допомогти вам проектувати темплейти ефективно:

Аналізуйте дизайн: При проектуванні темплейтів важливо аналізувати дизайн вашого сайту. Розгляньте компоненти, розташування блоків, різні стилі та елементи, які потрібно відобразити на сторінці. Розуміння дизайну допоможе вам створити відповідну структуру та розміщення елементів.

Розгляньте використання регіонів: Регіони дозволяють вам організувати розташування блоків та інших компонентів на сторінці. Розгляньте, які регіони вам потрібні для вашої теми, як вони пов'язані зі структурою сайту та як розмістити в них блоки та інші елементи.

Наслідування та включення: Використовуйте можливості наслідування та включення в шаблонах для створення більш складених структур. Визначте базові шаблони, які включають загальні елементи, такі як заголовок, навігація або підвал, та розширюйте їх у більш конкретних шаблонах.

Зберігайте шаблони зрозумілими та читабельними: При створенні шаблонів дотримуйтесь зрозумілого та читабельного коду. Використовуйте відступи, коментарі та відповідну назву файлів, щоб легко розрізняти та знайти потрібний шаблон.

Використовуйте змінні та фільтри: Використовуйте змінні та фільтри Twig для отримання та форматування даних, які ви відображаєте в шаблонах. Це дозволить вам легко керувати та модифікувати вміст.

Тестуйте тему на різних типах контенту: Перед запуском вашої теми, переконайтеся, що вона добре відображає різні типи контенту, такі як вузли, таксономія, блоки тощо. Перевірте, чи всі компоненти відображаються належним чином та чи не виникають проблеми з виглядом.

Не забувайте про адаптивність: Забезпечте, що ваша тема відображається належним чином на різних пристроях та розмірах екрану. Зверніть увагу на респонсивний дизайн, адаптивні зображення та правильне розташування елементів на різних пристроях.

Ці рекомендації допоможуть вам правильно проектувати темплейти в темі Drupal 8 та забезпечити ефективну розробку та відображення вашого сайту.

3.8.6. БІБЛІОТЕКИ

Бібліотеки (libraries) в темах Drupal 8 використовуються для організації та підключення зовнішніх ресурсів, таких як CSS-файли, JavaScript-файли, зображення та інші активи. Вони дозволяють контролювати завантаження та підключення ресурсів на сторінках вашого сайту.

Основні аспекти роботи з бібліотеками в темах Drupal 8:

Оголошення бібліотек: Бібліотеки оголошуються в файлі `yourtheme.libraries.yml` вашої теми. Ви можете вказати ім'я бібліотеки, залежності, шляхи до ресурсів та інші налаштування.

Наприклад:

```
example-library:
  version: 1.x
  css:
    theme:
      css/example.css: {}
  js:
    js/example.js: {}
```

Підключення бібліотек в темплейтах: Ви можете підключати оголошені бібліотеки в своїх темплейтах за допомогою функції `attach_library()`. Наприклад:

```
{{ attach_library('yourtheme/example-library') }}
```

Залежності бібліотек: Бібліотеки можуть мати залежності від інших бібліотек. Це дозволяє автоматично підключати всі необхідні ресурси. Ви можете вказати залежності для бібліотек в файлі `yourtheme.libraries.yml`. Наприклад:

```
example-library:
  version: 1.x
  css:
    theme:
      css/example.css: {}
  js:
    js/example.js: {}
  dependencies:
    - core/jquery
    - core/drupal
```

Шляхи до ресурсів: Ви можете вказати шляхи до CSS- та JS-файлів бібліотеки в файлі `yourtheme.libraries.yml`. Шляхи можуть бути відносними до папки теми або вказувати на зовнішні ресурси. Наприклад:

```
example-library:
 version: 1.x
 css:
  theme:
   css/example.css: {}
 js:
  js/example.js: {}
```

Оптимізація бібліотек: Drupal 8 має вбудовану функціональність для оптимізації підключення бібліотек. Ви можете використовувати налаштування Drupal, такі як агрегація та компресія ресурсів, для зменшення завантаження сторінок.

Бібліотеки дозволяють вам керувати підключенням ресурсів в темах Drupal 8, забезпечуючи ефективне та контрольоване завантаження CSS, JavaScript та інших активів на вашому сайті.

Використання CDN (Content Delivery Network) для підключення ресурсів у бібліотеках вашої теми в Drupal 8 дозволяє вам отримати певні переваги, такі як швидке завантаження ресурсів, зменшення навантаження на сервер та кешування ресурсів браузерами користувачів. Ось декілька кроків, які можуть допомогти вам використовувати CDN в бібліотеках вашої теми:

Оголошення CDN-залежностей: У файлі `yourtheme.libraries.yml` визначте ваші бібліотеки і встановіть CDN-залежності для CSS- та JS-файлів. Використовуйте ключ `remote` для вказання URL-адреси CDN. Наприклад:

```
example-library:
version: 1.x
css:
  theme:
    https://cdn.example.com/css/example.css: { type: external }
js:
  https://cdn.example.com/js/example.js: { type: external }
```

Підключення бібліотек з CDN у шаблонах: У вашому шаблоні використовуйте функцію `attach_library()` для підключення бібліотеки. Drupal автоматично визначить,

чи потрібно підключати ресурси з CDN або з локального сервера, в залежності від налаштувань.

{{ attach_library('yourtheme/example-library') }}

У Drupal 8 вага бібліотеки визначає порядок завантаження ресурсів, таких як CSS та JavaScript, на сторінці. Керування вагою бібліотек дозволяє вам контролювати, яка бібліотека буде завантажена першою, а яка - пізніше. Ось декілька способів керування вагою бібліотек в темах Drupal 8:

Керування вагою у файлі `yourtheme.libraries.yml`: У файлі `yourtheme.libraries.yml` можна встановити вагу для кожної бібліотеки, використовуючи ключ `weight`. Бібліотеки з меншою вагою завантажуються раніше, тоді як бібліотеки з більшою вагою завантажуються пізніше. Наприклад:

```
example-library:
  version: 1.x
  css:
    theme:
      css/example.css: {}
  js:
    js/example.js: {}
  weight: -10
```

Керування вагою за допомогою функції `attach_library()`: У вашому шаблоні ви можете використовувати функцію `attach_library()` для підключення бібліотеки та вказати вагу в якості параметра. Наприклад:

{{ attach_library('yourtheme/example-library', { weight: -10 }) }}

Керування вагою бібліотек дозволяє вам контролювати порядок завантаження ресурсів на сторінці. Використовуйте цей механізм, щоб забезпечити правильний порядок завантаження та виконання ресурсів вашої теми.

3.8.7. РОБОТА З CSS

При роботі з CSS в темі Drupal 8 потрібно враховувати кілька ключових аспектів. Ось детальний опис роботи з CSS в темах Drupal 8:

Структура CSS в темі:

У вашій темі має бути структура папок, що організовує ваш CSS-код. Зазвичай використовується наступна структура:

```
yourtheme/
├── css/
│   ├── base/
│   ├── components/
│   ├── layout/
│   └── theme/
└── yourtheme.info.yml
```

В папках `base`, `components`, `layout` та `theme` ви можете розміщувати відповідні CSS-файли, які відповідають за базовий стиль, компоненти, розміщення та стиль теми відповідно.

Оголошення CSS-файлів: Ви можете оголосити ваші CSS-файли в файлі `yourtheme.info.yml` вашої теми. Вказуйте шляхи до файлів CSS в розділі `stylesheets`. Наприклад:

```
stylesheets:
  all:
    - css/layout/layout.css: {}
    - css/components/button.css: {}
  theme:
    - css/theme/style.css: {}
```

Зазначені CSS-файли будуть автоматично підключені до вашої теми.

Використання CSS-класів: Використовуйте відповідні CSS-класи для елементів вашої теми. Drupal 8 використовує БЕМ-методологію (Блок, Елемент, Модифікатор),

але ви можете використовувати будь-яку іншу методологію або власні підходи до іменування класів.

Переоприділення CSS-стилів: Ви можете переоприділяти CSS-стилі, які визначені у модулях або ядрі Drupal, шляхом додавання власних CSS-правил у ваші файли CSS. Для цього використовуйте специфічність CSS-селекторів та правила.

Використання CSS-препроцесорів: Ви можете використовувати CSS-препроцесори, такі як Sass або Less, для зручного написання CSS-коду. Встановіть відповідний препроцесор, налаштуйте його і використовуйте його у вашій темі.

Компіляція та оптимізація CSS: Залежно від вашого налаштування, CSS може бути компільований та оптимізований в процесі розробки або на продакшен-сервері. Використовуйте інструменти, такі як Gulp, Grunt або Drupal-специфічні модулі, для компіляції та оптимізації вашого CSS-коду.

Відлагодження CSS: Використовуйте інструменти для відлагодження CSS, такі як розширення браузера для інспектування елементів, редактор стилів, або використовуйте вбудовані інструменти розробника браузера для відлагодження та тестування вашого CSS.

Важливо пам'ятати, що робота з CSS в темах Drupal 8 має бути добре структурованою та організованою, дотримуючись найкращих практик розробки CSS.

Агрегація та оптимізація CSS є важливою частиною оптимізації швидкості завантаження сторінок у темах Drupal 8. Давайте розглянемо ці процеси детальніше:

Агрегація CSS: Агрегація CSS полягає у збиранні всіх CSS-файлів в один файл. Це дозволяє зменшити кількість запитів до сервера, що поліпшує час завантаження сторінок. Drupal 8 надає можливість агрегувати CSS-файли за допомогою вбудованого механізму.

Ви можете ввімкнути агрегацію CSS, перейшовши до "Configuration" -> "Development" -> "Performance" в адміністративному інтерфейсі Drupal. У розділі "CSS aggregation" виберіть опцію "Aggregate CSS files". Після цього всі CSS-файли, оголошені в вашій темі, будуть агреговані в один файл.

Оптимізація CSS: Оптимізація CSS включає в себе різні техніки, які допомагають зменшити розмір CSS-файлів та покращити швидкість завантаження. Ось декілька рекомендацій:

Мініфікація: Застосуйте мініфікацію до CSS-коду, що полягає у видаленні непотрібних пробілів, коментарів та зменшенні кількості символів. Це можна зробити

вручну або за допомогою спеціальних інструментів, таких як Grunt або Gulp з відповідними плагінами.

Компресія: Використовуйте компресію для зменшення розміру CSS-файлів. Зазвичай використовується gzip-компресія, яка дозволяє стиснути файли перед їх відправкою на клієнтський браузер.

Видалення непотрібних стилів: Перевірте свої CSS-файли на наявність непотрібних стилів або стилів, які не використовуються на сторінці. Видалення непотрібного CSS допоможе зменшити розмір файлу.

Кешування CSS: Для поліпшення швидкості завантаження сторінок можна використовувати кешування CSS-файлів на стороні браузера. За допомогою налагоджень кешування веб-сервера або використовуючи спеціальні модулі Drupal, такі як "Advanced CSS/JS Aggregation", ви можете налаштувати кешування CSS-файлів на довший період, щоб клієнтські браузери могли зберігати копії файлів і не завантажувати їх знову при кожному запиті.

Важливо експериментувати, тестувати та відстежувати результати для забезпечення оптимальної агрегації та оптимізації CSS-файлів у вашій Drupal 8 темі.

3.8.8. РОБОТА З JAVASCRIPT

При роботі з JavaScript (JS) в темах Drupal 8 є кілька ключових аспектів, які варто враховувати. Ось детальний опис роботи з JS в темах Drupal 8:

Оголошення JS-файлів:

Ви можете оголосити ваші JS-файли в файлі `yourtheme.libraries.yml` вашої теми. Вказуйте шляхи до файлів JS в розділі `javascript`. Наприклад:

```
javascript:
  js/custom.js: {}
```

Зазначені JS-файли будуть автоматично підключені до вашої теми.

Підключення JS-файлів в темплейтах:

Ви можете підключати оголошені JS-файли в своїх темплейтах за допомогою функції `attach_library()`. Наприклад:

```
{{ attach_library('yourtheme/custom-js') }}
```

Використання бібліотек JavaScript:

Drupal 8 дозволяє використовувати сторонні бібліотеки JavaScript, такі як jQuery, React, Vue.js тощо, в вашій темі. Ви можете оголосити залежності в файлі `yourtheme.libraries.yml` та підключити їх за допомогою `attach_library()`. Наприклад:

```
dependencies:
 - core/jquery
 - core/drupalSettings
```

Робота з глобальним об'єктом `Drupal`:

Drupal 8 надає глобальний об'єкт `Drupal`, який містить корисні функції та методи для роботи з Drupal API. Ви можете використовувати `Drupal.behaviors` для підключення функцій, які будуть виконуватися при завантаженні сторінки або після AJAX-оновлення. Наприклад:

```
(function ($, Drupal) {
 Drupal.behaviors.myBehavior = {
  attach: function (context, settings) {
   // Ваш код JavaScript
  }
 };
})(jQuery, Drupal);
```

Керування порядком завантаження JS-файлів:

Ви можете вказати порядок завантаження JS-файлів, встановивши значення `weight` у файлі `yourtheme.libraries.yml`. Більша вага вказує на більш пізнє завантаження файлу. Наприклад

```
javascript:
 js/custom.js:
  weight: 10
 js/other.js:
  weight: 20
```

Файл `other.js` буде завантажено після `custom.js`.

Оптимізація та компресія JS:

Використовуйте інструменти, такі як Gulp або Grunt з відповідними плагінами, для мініфікації, компресії та оптимізації вашого JS-коду. Це допоможе зменшити розмір файлів та покращити швидкість завантаження сторінок.

Відлагодження JS:

Використовуйте розширення браузера для відлагодження JavaScript, такі як інструменти розробника Chrome або Firefox, для перевірки та налагодження вашого JS-коду.

При роботі з JS в темах Drupal 8 важливо ретельно організовувати код, використовувати найкращі практики, дотримуватись принципів модульності та забезпечувати сумісність з іншими модулями та темами.

Агрегація та оптимізація JavaScript (JS) є важливими аспектами оптимізації швидкості завантаження сторінок у темах Drupal 8. Давайте розглянемо ці процеси детальніше:

Агрегація JS:

Агрегація JS включає збір всіх JS-файлів в один файл з метою зменшення кількості запитів до сервера. Це поліпшує час завантаження сторінок. Ви можете ввімкнути агрегацію JS, перейшовши до "Configuration" -> "Development" -> "Performance" в адміністративному інтерфейсі Drupal. У розділі "JS aggregation" виберіть опцію "Aggregate JavaScript files". Після цього всі JS-файли, оголошені в вашій темі, будуть агреговані в один файл.

Оптимізація JS:

Оптимізація JS включає різні техніки для зменшення розміру JS-файлів та покращення швидкості завантаження. Ось кілька рекомендацій:

Мініфікація: Використовуйте мініфікацію для стиснення JS-коду, видаляючи непотрібні пробіли, коментарі та зайві символи. Це можна зробити вручну або за допомогою спеціальних інструментів, таких як UglifyJS або Terser.

Компресія: Застосуйте компресію до JS-файлів для зменшення їх розміру. Використовуйте gzip-компресію на веб-сервері, щоб стиснути файли перед їх відправкою на клієнтський браузер.

Виключення непотрібного коду: Перевірте ваш JS-код на наявність непотрібного або не використовуваного коду. Видаліть зайві функції, змінні або код, який не використовується на сторінці.

Кешування JS:

Для покращення швидкості завантаження сторінок можна використовувати кешування JS-файлів на стороні браузера. За допомогою налаштувань кешування веб-сервера або використовуючи спеціальні модулі Drupal, такі як "Advanced CSS/JS Aggregation", ви можете налаштувати кешування JS-файлів на довший період, щоб клієнтські браузери зберігали копії файлів і не завантажували їх знову при кожному запиті.

Відлагодження JS:

Використовуйте інструменти для відлагодження JavaScript, такі як розширення браузера для розробників, такі як Chrome DevTools або Firefox Developer Tools, для перевірки та налагодження вашого JS-коду.

Важливо експериментувати, тестувати та відстежувати результати для забезпечення оптимальної агрегації та оптимізації JS-файлів у вашій Drupal 8 темі.

3.8.9. PREPROCESS-ФУНКЦІЇ ТА THEME SUGGESTIONS

Препроцес функції є важливою частиною тем розробки в Drupal 8. Вони дозволяють вам маніпулювати та модифікувати дані перед їх відображенням у темплейтах. Ось детальний опис препроцес функцій в темах Drupal 8:

Структура препроцес функцій:

Препроцес функції зазвичай розміщуються у файлі теми з назвою `yourtheme.theme`, де `yourtheme` - назва вашої теми. Цей файл повинен знаходитись в кореневій папці вашої теми.

Синтаксис препроцес функцій:

Препроцес функції у Drupal 8 мають певний синтаксис. Ось загальний синтаксис препроцес функції:

```
/**
 * Implements hook_preprocess_HOOK() for THEMENAME.
 */
function yourtheme_preprocess_HOOK(&$variables) {
  // Ваш код тут
}
```

У цьому прикладі `HOOK` - це назва темплейту або елемента, до якого ви хочете застосувати препроцес функцію. Наприклад, `page`, `block`, `node`, `field`, тощо.

Препроцес функції отримують змінні як параметр та дозволяють вам змінювати їх значення. Зазвичай це асоціативний масив `$variables`, який містить дані, що передаються в темплейт. Ви можете змінювати ці значення або додавати нові змінні до масиву.

Приклади препроцес функцій: Ось кілька прикладів препроцес функцій, які ви можете використовувати:

Препроцес функція для зміни змінної `$title` на сторінці:

```
function yourtheme_preprocess_page(&$variables) {
  $variables['title'] = 'Новий заголовок сторінки';
}
```

Препроцес функція для додавання нової змінної `$custom_variable`:

```
function yourtheme_preprocess_page(&$variables) {
  $variables['custom_variable'] = 'Значення нової змінної';
}
```

Препроцес функція для додавання класу до елемента:

```
function yourtheme_preprocess_block(&$variables) {
   $variables['attributes']['class'][] = 'custom-class';
 }
```

Кешування препроцес функцій:

Препроцес функції можуть бути кешовані Drupal, щоб покращити продуктивність. Ви можете встановити кешування за допомогою параметру `cache` для масиву змінних. Наприклад:

```
function yourtheme_preprocess_page(&$variables) {
  $variables['#cache']['max-age'] = 3600; // Кешувати на 1 годину
 }
```

Це допомагає уникнути виконання препроцес функції при кожному запиті, якщо дані не змінюються.

Препроцес функції дозволяють вам змінювати та модифікувати дані перед їх відображенням в темплейтах теми Drupal 8. Вони є потужним інструментом для налаштування вигляду вашого сайту.

3.8.10. ОСОБЛИВОСТІ РОЗРОБКИ ТЕМ, ВІДЛАГОДЖЕННЯ ТЕМ

При створенні тем для Drupal 8 є кілька ключових аспектів, на які варто звертати увагу. Ось детальний опис того, на що варто звертати увагу при розробці тем для Drupal 8:

HTML структура:

Правильна HTML структура важлива для забезпечення доступності та зручності використання сайту. Використовуйте семантичні теги HTML для розмітки різних частин сайту (наприклад, `<header>`, `<nav>`, `<main>`, `<footer>` тощо) і стежте за відповідними практиками.

CSS і дизайн:

Розробка тем включає стилізацію вашого сайту за допомогою CSS. Зверніть увагу на такі аспекти:

Чистий і організований CSS код.

Використання каскадних стилів і правильне вкладення для забезпечення легкості редагування і підтримки.

Використання адаптивного дизайну для забезпечення коректного відображення вашого сайту на різних пристроях та розмірах екрану.

Використання темплейтів:

Drupal 8 використовує систему темплейтів для відображення контенту. Ви можете використовувати темплейти для кастомізації різних типів контенту, блоків, форм тощо. Ознайомтесь зі структурою та можливостями темплейтів Drupal 8, включаючи Twig-темплейти, та використовуйте їх для досягнення потрібного вигляду вашого сайту.

Функціональність:

Враховуйте функціональні аспекти вашої теми, такі як:

Підтримка модулів: Деякі модулі можуть надавати додаткові функціональні можливості для тем. Розгляньте можливості інтеграції з такими модулями та їх практичне використання.

Підтримка мобільних пристроїв: Забезпечте, щоб ваша тема була адаптивною та добре працювала на різних пристроях.

SEO оптимізація: Використовуйте правильні теги HTML, метатеги та оптимізуйте URL-адреси для покращення SEO вашого сайту.

Кросс-браузерна сумісність:

Перевірте, що ваша тема працює коректно на різних веб-браузерах, таких як Chrome, Firefox, Safari, Edge та Internet Explorer. Переконайтеся, що ваші стилі та функціональність працюють без проблем на різних браузерах.

Тестування:

Перед релізом вашої теми переконайтеся, що ви провели достатнє тестування, включаючи перевірку відображення на різних пристроях, перевірку сумісності з різними версіями Drupal та відповідність функціональним вимогам.

Документація:

Добре задокументована тема полегшує її розуміння та редагування в майбутньому. Забезпечте наявність детальної документації, яка описує особливості, налаштування та використання вашої теми.

Оновлення:

Зверніть увагу, що Drupal може випускати нові версії, які можуть впливати на вашу тему. Підтримуйте вашу тему, оновлюючи її відповідно до нових вимог та функціональності Drupal.

Враховуючи ці аспекти, ви зможете розробити високоякісну тему для Drupal 8, яка буде функціональною, естетичною та зручною для користувачів.

Питання для самоконтролю:

1. Що таке тема?
2. Як працює Twig?
3. Що таке препроцес-функції?
4. Як додати CSS та JS до теми?
5. Як оптимізувати CSS та JS?

3.9. РОЗРОБКА БЕЗПЕЧНОГО КОДУ

Написання безпечного коду є надзвичайно важливим аспектом розробки в Drupal 8. Важливість безпечного коду полягає у забезпеченні захисту вашого веб-сайту та його користувачів від різних видів атак та вразливостей. Ось деякі важливі причини, чому безпечний код є настільки важливим:

Захист від зловживання. Незахищений код може бути вразливим для атак, таких як внедрення зловмисного коду, XSS (Cross-Site Scripting), SQL-ін'єкції та інших атак, що можуть призвести до викрадення конфіденційної інформації, порушення безпеки даних або недоступності сайту. Написання безпечного коду дозволяє запобігати цим видам зловживань та забезпечує відповідний рівень безпеки.

Захист конфіденційної інформації. Безпечний код допомагає захистити конфіденційну інформацію, таку як паролі, особисті дані користувачів, фінансова інформація тощо. Правильне оброблення, зберігання та передача цих даних у безпечному форматі допомагає запобігти несанкціонованому доступу та витоку інформації.

Забезпечення надійності та стабільності. Незахищений код може призвести до помилок, відмов та інших проблем, які можуть вплинути на надійність та стабільність вашого сайту. Безпечний код зменшує ризик виникнення таких проблем та допомагає підтримувати ваш веб-сайт в робочому стані.

Виконання нормативних вимог. Багато веб-сайтів, зокрема ті, які працюють з особистими даними або фінансовою інформацією, повинні відповідати різним нормативним вимогам щодо безпеки, таким як GDPR (Загальне регламентування про захист даних) або PCI DSS (Стандарт безпеки даних галузі платіжних карток). Написання безпечного коду допомагає відповідати цим нормативним вимогам та забезпечує дотримання необхідних стандартів безпеки.

Отже, написання безпечного коду в Drupal 8 є критичним для забезпечення захисту вашого веб-сайту, конфіденційності даних та надійності. Використовуйте найкращі практики безпеки, перевіряйте ваш код на вразливості та виконуйте рекомендації безпеки Drupal для створення надійних та безпечних веб-сайтів.

3.9.1. СТАНДАРТИ КОДУВАННЯ

Стандарти кодування в Drupal 8, такі як Drupal Coding Standards і PSR (PHP Standards Recommendations), є набором рекомендацій і правил, які допомагають розробникам створювати чистий, читабельний і однорідний код. Ось детальний опис стандартів кодування для Drupal 8 та їх важливість:

Drupal Coding Standards: Drupal Coding Standards - це набір правил і рекомендацій для написання коду в Drupal 8. Ці стандарти включають розміщення дужок, використання відступів, неймспейсів, коментування коду, назви змінних та функцій, і багато іншого. Використання Drupal Coding Standards допомагає створювати єдиний стиль коду для всього проекту, полегшує читання та розуміння коду, а також полегшує співпрацю з іншими розробниками.

PSR (PHP Standards Recommendations): PSR - це набір рекомендацій для PHP, розроблений PHP Framework Interop Group (FIG). Вони мають на меті стандартизацію і однорідність в PHP-середовищі. Наприклад, PSR-1 стосується базових стандартів кодування, PSR-4 стосується автозавантажувачів класів, PSR-7 стосується HTTP-повідомлень, і т.д. Використання PSR у Drupal 8 допомагає полегшити співпрацю з іншими PHP-розробниками і полегшує переносимість коду між проектами.

Важливість стандартів кодування:

- Читабельність: Чистий і однорідний код полегшує читання, розуміння і обслуговування коду. Інші розробники будуть легше співпрацювати з вами, коли вони можуть легко зрозуміти і налагоджувати ваш код.
- Консистентність: Використання стандартів кодування допомагає створити єдиний стиль коду для всього проекту. Це зробить ваш код більш консистентним, забезпечить однорідність та зручність для розробників, які працюватимуть з вашим кодом.
- Переносимість: Дотримання стандартів кодування допомагає покращити переносимість вашого коду. Ви зможете легко перенести частини коду з одного проекту в інший без необхідності вносити значні зміни в стиль коду.
- Інструменти та автоматична перевірка: Використання стандартів кодування дозволяє використовувати автоматичні інструменти, такі як linters та code sniffer, для перевірки вашого коду на відповідність цим стандартам. Це допомагає виявляти та виправляти потенційні проблеми з кодом на ранніх етапах розробки.

Загалом, використання стандартів кодування в Drupal 8 допомагає створити високоякісний, читабельний, однорідний та безпечний код, що полегшує співпрацю з іншими розробниками, забезпечує переносимість та зручність обслуговування вашого проекту.

При розробці в Drupal 8 є кілька способів контролювати та забезпечувати дотримання стандартів коду. Ось кілька з них:

Code Sniffer: Використання інструменту Code Sniffer дозволяє перевіряти ваш код на відповідність стандартам кодування. В Drupal 8 існує набір правил для Code Sniffer, які відповідають Drupal Coding Standards. Ви можете використовувати команду `phpcs` або відповідні плагіни для вашої редактора коду для автоматичної перевірки вашого коду.

Drupal Code Review: Drupal Code Review (DCR) - це онлайн-інструмент, який дозволяє завантажити ваш код та отримати рекомендації щодо відповідності Drupal Coding Standards. Він пропонує докладний аналіз вашого коду та надає рекомендації щодо виправлення помилок або невідповідностей стандартам.

IDE або редактор коду: Багато редакторів коду та інтегрованих середовищ розробки (IDE), таких як PhpStorm, Visual Studio Code, Sublime Text, мають плагіни або розширення, які допомагають перевіряти стандарти кодування під час розробки. Вони можуть надавати вказівки та попередження про невідповідності стандартам коду.

Використання статичного аналізатора коду: Статичні аналізатори коду, такі як PHPStan або Psalm, допомагають виявляти потенційні помилки, недоліки та невідповідності стандартам кодування. Вони використовують різні алгоритми та правила, щоб перевірити ваш код на можливі проблеми.

Ревізія коду та піар-рев'ю: Організуйте регулярну ревізію коду внутрішньої команди або спільноти розробників Drupal. Інші розробники можуть перевірити ваш код і надати цінні поради щодо відповідності стандартам.

Загалом, важливо не тільки використовувати інструменти для перевірки стандартів коду, але й свідомо дотримуватися стандартів під час написання коду. Розробники повинні бути ознайомлені зі стандартами кодування Drupal 8 та зусиллями спільноти для створення чистого, читабельного та однорідного коду.

3.9.2. ФІЛЬТРАЦІЯ ВХІДНОГО ПОТОКУ ДАНИХ

Фільтрація вхідних даних в Drupal 8 є важливим кроком для забезпечення безпеки вашого сайту та захисту від різних видів атак, таких як XSS (Cross-Site Scripting) або SQL-ін'єкція. Ось кілька рекомендацій щодо фільтрації вхідних даних в Drupal 8:

Використання фільтрів фільтрації: Drupal 8 надає вбудовані функції фільтрації для очищення та форматування вхідних даних. Наприклад, функція `Xss::filter()` використовується для очищення вхідних даних від потенційно небезпечних HTML та JavaScript коду. Використовуйте ці функції для фільтрації даних перед їх використанням.

Використання форм API: Коли створюєте форми в Drupal 8, використовуйте вбудовану систему форм API. Вона автоматично захищає вхідні дані від XSS атак шляхом очищення та екранування даних, переданих через форму.

3. Використання функцій перевірки та очищення: Drupal 8 надає функції для перевірки та очищення різних типів даних. Наприклад, для перевірки користувацького вводу використовуйте функцію `Drupal\Component\Utility\EmailValidator::isValid()`, а для очищення URL-адрес використовуйте функцію `Drupal\Component\Utility\UrlHelper::filterBadProtocol()`. Використовуйте ці функції для забезпечення правильної обробки та фільтрації вхідних даних.

4. Валідація форм: Крім фільтрації даних, використовуйте валідацію форм для перевірки правильності та безпеки введених користувачем даних. Використовуйте вбудовану систему валідації форм в Drupal 8 для перевірки та очищення даних, що надходять через форму.

5. Перевірка прав доступу: Крім фільтрації даних, також важливо перевіряти права доступу користувачів до вхідних даних. Виконуйте перевірку на права доступу та авторизацію, перш ніж обробляти вхідні дані та виконувати певні дії на основі цих даних.

Загалом, фільтрація вхідних даних в Drupal 8 має на меті очищення та форматування даних, щоб запобігти можливим атакам та забезпечити безпеку вашого сайту. Використовуйте вбудовані функції та механізми Drupal 8 для ефективної фільтрації вхідних даних та запобігання потенційним загрозам безпеці.

Filter API в Drupal 8 є механізмом для обробки та форматування текстового вмісту, зокрема фільтрації HTML-коду та інших потенційно небезпечних елементів.

Цей API дозволяє вам застосовувати різні фільтри до тексту, щоб забезпечити безпеку та відповідність вмісту встановленим правилам.

Основні компоненти Filter API включають:

Фільтри: Фільтри визначають, які зміни вносяться в текстовий вміст. В Drupal 8 вбудовані різні фільтри, такі як "Basic HTML" (основний HTML), "Limit HTML tags" (обмеження HTML-тегів), "Convert URLs to links" (конвертування URL-адрес в посилання) та багато інших. Ви також можете створювати власні фільтри, які відповідають вашим потребам.

Фільтраційні формати: Фільтраційні формати - це збірки фільтрів, які використовуються для обробки текстового вмісту в певному контексті. У Drupal 8 є попередньо визначені формати, такі як "Basic HTML" (основний HTML), "Full HTML" (повний HTML), "Filtered HTML" (фільтрований HTML) та "Plain text" (простий текст). Ви також можете створювати власні формати для використання в специфічних випадках.

Обробка тексту: Під час обробки тексту Filter API використовує послідовне застосування фільтрів до тексту. Кожен фільтр може модифікувати вміст шляхом видалення, заміни або додавання певних елементів. Обробка тексту відбувається під час збереження або виведення вмісту.

Налаштування фільтраційних форматів: Ви можете настроїти фільтраційні формати в адміністративному інтерфейсі Drupal, де ви можете вибрати, які фільтри використовуються для кожного формату, а також встановити додаткові настройки для кожного фільтра. Наприклад, ви можете встановити список дозволених тегів та атрибутів HTML для певного формату.

Власні фільтри та розширення: Filter API надає можливість створювати власні фільтри та розширювати функціональність існуючих фільтрів. Ви можете створювати фільтри за допомогою плагінів Drupal і додавати їх до ваших фільтраційних форматів. Це дозволяє вам забезпечити спеціалізовану обробку тексту відповідно до вашого власного контексту.

Filter API в Drupal 8 дозволяє контролювати та обробляти вміст, що вводиться користувачами, забезпечуючи безпеку та відповідність стандартам. Використовуйте фільтри та фільтраційні формати для забезпечення правильної обробки та відображення вмісту на вашому сайті.

3.9.3. ЗАХИСТ ВІД SQL INJECTION

Захист від SQL-ін'єкцій є критично важливим аспектом безпеки веб-додатків, включаючи Drupal 8. Друпал 8 використовує підготовлені запити та інші вбудовані механізми для запобігання SQL-ін'єкціям. Ось кілька рекомендацій щодо захисту від SQL-ін'єкцій в Drupal 8:

Використання підготовлених запитів: Використання підготовлених запитів дозволяє передавати параметри до SQL-запиту окремо від самого запиту. Це дозволяє базі даних правильно інтерпретувати передані значення і запобігає внедренню шкідливого SQL-коду через вхідні дані. В Drupal 8 ви можете використовувати методи підготовлених запитів, такі як `select`, `update`, `insert`, `delete` з об'єктом `DatabaseConnection` або `EntityQuery`.

Валідація та очищення вхідних даних: Перевіряйте та очищайте вхідні дані перед використанням їх в SQL-запитах. Використовуйте функції, такі як `Database::escapeString()` або `Connection::escapeLiteral()`, для екранування спеціальних символів в рядках перед їх включенням до запиту. Також перевіряйте вхідні дані на правильність та типи перед їх використанням.

Використання ORM (Object-Relational Mapping): У Drupal 8 ви можете використовувати ORM, такий як Entity API, для звертання до бази даних. ORM автоматично генерує безпечні SQL-запити на основі ваших об'єктів та їх властивостей, що дозволяє уникнути SQL-ін'єкцій. Використовуйте ORM там, де це можливо, замість ручного складання SQL-запитів.

Використання модулів безпеки: Drupal 8 має декілька модулів безпеки, таких як "Security Kit" і "Paranoia", які надають додаткові заходи безпеки для запобігання SQL-ін'єкціям. Розгляньте використання цих модулів для посилення захисту вашого сайту.

Регулярні оновлення: Важливо регулярно оновлювати Drupal та всі його модулі до останніх версій, оскільки оновлення можуть включати покращені заходи безпеки та виправлення вразливостей, пов'язаних з SQL-ін'єкціями.

Пам'ятайте, що захист від SQL-ін'єкцій - це комплексний підхід, який включає правильне використання методів підготовлених запитів, валідацію та очищення даних, використання ORM та виконання регулярних оновлень Drupal та його модулів.

3.9.4. BEST PRACTICES FOR WRITING SECURE CODE

Написання безпечного коду в Drupal 8 має велике значення для забезпечення захисту вашого сайту та його користувачів. Ось кілька кращих практик, які варто враховувати при розробці безпечного коду на Drupal 8:

Запобігайте XSS (Cross-Site Scripting) атакам:

- Всі вхідні дані, які виводяться на сторінках, повинні бути корректно очищені та екрановані.
- Використовуйте вбудовані функції фільтрації, такі як `Xss::filter()` або `Html::escape()`, для очищення вхідних даних від небезпечних HTML тегів та скриптів.
- Використовуйте контекстно-залежне екранування, наприклад, `{{ content|raw }}`, лише якщо ви впевнені, що вміст безпечний.

Уникайте SQL-ін'єкцій:

- Використовуйте підготовлені запити, такі як методи `select`, `update`, `insert`, `delete` з об'єктом `DatabaseConnection` або `EntityQuery`, щоб передавати параметри до SQL-запитів окремо від самого запиту.
- Використовуйте валідацію та очищення вхідних даних перед використанням їх в SQL-запитах.
- Не довіряйте вхідним даним користувачів і не вставляйте їх безпосередньо в SQL-запити.

Забезпечте коректну обробку файлів:

- Перевіряйте типи файлів та їх розміри перед завантаженням або використанням.
- Забезпечте, щоб завантажені файли були збережені у безпечному місці та не могли виконати код на сервері.

Перевіряйте права доступу:

- Перевіряйте, чи мають користувачі необхідні права доступу до ресурсів, перед тим як надавати їм доступ до важливих функцій або даних.

Використовуйте модулі безпеки:

- В Drupal 8 є різні модулі безпеки, такі як "Security Kit", "Paranoia", "Password Policy" та інші. Використовуйте ці модулі для посилення безпеки вашого сайту та дотримання рекомендацій безпеки.

Регулярно оновлюйте Drupal та його модулі:

- Важливо регулярно оновлювати Drupal та всі встановлені модулі до останніх версій, оскільки оновлення можуть включати виправлення вразливостей та покращені заходи безпеки.

Ревізія коду та тестування:

- Проводьте регулярну ревізію коду, щоб виявити можливі проблеми безпеки та вразливості.
- Виконуйте тестування на безпеку, включаючи тестування на переповнення буфера, XSS, CSRF та інші потенційні загрози.

Ці кращі практики допоможуть вам створити безпечний код для вашого Drupal 8 сайту та захистити його від можливих загроз безпеки. Пам'ятайте, що безпека - постійний процес, тому важливо постійно оновлювати свої знання та практики безпеки і вдосконалювати свої проекти залежно від нових відомостей про потенційні загрози.

ПЕРЕЛІК ЛІТЕРАТУРИ

1. Ullman, Larry. "PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide." Peachpit Press, 2017.
2. Welling, Luke and Thomson, Laura. "PHP and MySQL Web Development." Addison-Wesley Professional, 2016.
3. Lerdorf, Rasmus and Tatroe, Kevin. "Programming PHP." O'Reilly Media, 2013.
4. Sipos, Daniel. "Drupal 8 Module Development: Build and Customize Drupal 8 Modules and Extensions Efficiently." Packt Publishing, 2016.
5. Glaman, Matt. "Drupal 8 Development Cookbook." Packt Publishing, 2016.
6. Chumley, Chaz. "Drupal 8 Theming with Twig." Packt Publishing, 2016.
7. Finklea, Ben. "Drupal 8 SEO." O'Reilly Media, 2017.
8. Nixon, Robin. "Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5." O'Reilly Media, 2014.
9. Lockhart, Josh. "Modern PHP: New Features and Good Practices." O'Reilly Media, 2015.
10. Sklar, David and Trachtenberg, Adam. "PHP Cookbook." O'Reilly Media, 2014.
11. Powers, David. "PHP Solutions: Dynamic Web Design Made Easy." Friends of ED, 2014.
12. Horton, Matt. "Beginning PHP and MySQL: From Novice to Professional." Apress, 2010.
13. Turnbull, Matthew. "Drupal 8 Explained: Your Step-by-Step Guide to Drupal 8." OSTraining, 2016.
14. Mercer, Todd. "The Joy of PHP: A Beginner's Guide to Programming Interactive Web Applications with PHP and mySQL." CreateSpace Independent Publishing Platform, 2012.
15. Freeman, Adam, et al. "Pro Drupal 7 Development." Apress, 2010.
16. Melançon, Benjamin et al. "The Definitive Guide to Drupal 7." Apress, 2011.
17. Summerfield, Mark. "Programming in Python 3: A Complete Introduction to the Python Language." Addison-Wesley Professional, 2009.
18. Lerdorf, Rasmus, et al. "Programming PHP." O'Reilly Media, 2013.
19. Butterworth, Adam, et al. "PHP and MySQL Web Development All-in-One Desk Reference for Dummies." For Dummies, 2013.
20. Yank, Kevin. "PHP and MySQL: Novice to Ninja." SitePoint, 2012.

21. Soh, Keong. "PHP Advanced and Object-Oriented Programming: Visual QuickPro Guide." Peachpit Press, 2012.
22. Lengstorf, Jason, and David Powers. "PHP Master: Write Cutting-Edge Code." SitePoint, 2011.
23. Doyle, Matt. "Beginning PHP 6, Apache, MySQL 6 Web Development." Apress, 2009.
24. Kravets, Andrey. "Learning PHP, MySQL, JavaScript, CSS & HTML5: A Step-by-Step Guide to Creating Dynamic Websites." O'Reilly Media, 2014.
25. Hahn, Jesse, et al. "Professional WordPress: Design and Development." Wrox, 2015.
26. Kosmaczewski, Konstantin. "Swift Programming for Beginners." Packt Publishing, 2015.
27. Lefebvre, Chris, et al. "Drupal 8 Explained: Your Step-by-Step Guide." OSTraining, 2016.
28. Tisdall, James. "Beginning Perl." Wiley, 2012.
29. Yank, Kevin, et al. "PHP & MySQL: Novice to Ninja." SitePoint, 2017.

ЗМІСТ

Вступ……..2
Глосарій…….3
1. Загальні поняття веб-програмування…….5
1.1. Поняття про веб-програмування. Бекенд та фронтенд розробка……5
1.2. Класифікація мов програмування та розмітки…….9
1.2.1. Мова HTML та її особливості…….10
1.2.2. Мова CSS та її особливості…….15
1.2.3. Мова Javascript та її особливості…….21
1.2.4. Мова PHP та її особливості…….27
1.2.5. Мова запитів реляційних баз даних SQL……...35
1.3. Локальне та серверне оточення……44
1.3.1. Стеки LAMP та WAMP……46
1.3.2. Веб-сервери Apache та Nginx……..48
1.3.3. Інтерпретатор мови PHP. Особливості версій PHP 7.x та 5.x……...52
1.3.4. Сервери MySQL / MariaDB…….57
1.4. Засоби відлагодження. Дебагери…….62
1.5. Огляд систем інтегрованої розробки веб-додатків…….67
2. Основи роботи з CMF Drupal (Site building).....70
2.1. Загальна характеристика системи……70
2.2. Інсталяція та базові налаштування……..72
2.2.1. Системні вимоги……...75
2.2.2. Налаштування локального оточення……..76
2.2.3. Інсталяція……..79
2.2.4. Використання інтерпретатора Drush……..80
2.2.5. Встановлення модулів та тем……..85
2.2.6. Базові налаштування……90
2.3. Сутності (Entities) в Drupal……...94
2.3.1. Поняття сутності. Основні сутності в Drupal……95
2.3.2. Типи контенту, поля, налаштування форм і режимів відображення…...97
2.3.3. Таксономія та її використання……..103
2.3.4. Профілі користувачів та їх налаштування……...109
2.3.5. Блоки та їх налаштування……..113
2.3.6. Меню та їх налаштування……..117

2.4. Конструктор запитів Views……121
2.4.1. Загальні відомості про конструктор запитів Views……122
2.4.2. Принципи використання Views……123
2.4.3. Побудова сторінок та блоків з використанням Views……125
2.4.4. Використання режимів відображення у Views……130
2.4.5. Статичні та динамічні фільтри у Views……135
2.4.6. Налаштування агрегації даних у Views……141
2.5. Керування правами доступу……...147
2.5.1. Структура облікових записів користувачів……..148
2.5.2. Користувацькі ролі……152
2.5.3. Права доступу……156
2.5.4. Підходи до організації розподіленого доступу до контенту……..158
2.5.5. Засоби для розширення функціоналу управління доступом…….162
2.5.6. Використання облікових записів та ролей……...165
2.6. Управління конфігураціями в Drupal……167
2.6.1. Поняття конфігурації……168
2.6.2. Менеджер конфігурацій та його використання……...170
2.6.3. Експорт конфігурацій……173
2.6.4. Імпорт конфігурацій……...174
2.6.5. Використання Drush для роботи з конфігураціями……175
2.6.6. Типові рішення для управління конфігураціями……176
2.7. Рішення для безпеки та захисту даних в Drupal……...179
2.7.1. Загальні правила щодо забезпечення безпеки і захисту даних……..180
2.7.2. Системне логування……...183
2.7.3. Управління правами доступу до файлової системи……185
2.7.4. Потенційно небезпечні елементи конфігурацій……..187
2.7.5. Оновлення та патчі……188
2.7.6. Типові методи виявлення та нейтралізації вразливостей……...191
3. Основи розробки веб-програмування в CMF Drupal……193
3.1. Поняття про модулі та теми. Стандарти кодування. Відлагодження…….194
3.2. Розробка модулів……201
3.2.1 Структура файлів та директорій модуля……...202
3.2.2. Структура info.yml……205
3.2.3. Роути……207

3.2.4. Контроллери……213
3.2.5. Плагіни……218
3.2.6. Форми……..229
3.2.7. Сервіси……234
3.2.8. Права доступу……238
3.3. Хуки та події……240
3.4. Робота з сутностями. Entity API……245
3.5. Робота з базою даних. Database API……..253
3.6. Робота з Configuration API……..259
3.7. Робота з State API……262
3.8. Розробка тем……265
3.8.1. Структура теми……...266
3.8.2. Структура .info.yml……269
3.8.3. Структура .theme……270
3.8.4. Технологія twig……...272
3.8.5. Темплейти та їх використання……..273
3.8.6. Бібліотеки……276
3.8.7. Робота з CSS……280
3.8.8. Робота з JavaScript……..282
3.8.9. Preprocess-функції та theme suggestions……...285
3.8.10. Особливості розробки тем, відлагодження тем……287
3.9. Розробка безпечного коду……...289
3.9.1. Стандарти кодування……..290
3.9.2. Фільтрація вхідного потоку даних……292
3.9.3. Захист від SQL injection……294
3.9.4. Best practices for writing secure code……..295
Перелік літератури……...297