УДК 004.8

Ячменьов Я. О., магістрант
Панаріна І. В., канд. техн. наук., доцент
Державний університет «Житомирська політехніка»

СИСТЕМА ІНТЕҐРАЦІЇ MAGENTO 2 У WORDPRESS

Із розвитком мережі інтернет та просуванням її до широкого загалу різкими темпами почала прогресувати електронна комерція (надалі ЕК), що торкнулася всіх куточків нашого життя в наступних своїх проя-вах: електронний обмін інформацією, електронний рух капіталу, електронна торгівля, електронні гроші, електронний маркетинг, електронний банкінг, електронні страхові послуги тощо. Досить цікавою складовою ЕК є електронні магазини. Нині їхній розвиток та потреби сучасного споживача сприяють виникненню великих фреймворків, систем керуванням вмістом (надалі СКВ), що мають розвинену спільноту програмістів у всьому світі.

Після ретельного аналізу ситуації з відсутністю рішення для забезпечення електронного магазину на платформі Magento 2 багатофунк-ціональним блоґом було поставлене завдання розробки плаґіну під WordPress, що дозволить ефективно поєднати вміст системи Magento 2 у блозі, створеному на WordPress, матиме подійно-орієнтовану архітек-туру для легшої інтеґрації із плаґінами інших WordPress розробників.

Основною частиною системи є ядро інтеґрації Magento 2 у WordPress у вигляді статичного класу «M2I_External». Клас «M2I_External» виконує не тільки роль монолітного ядра інтеґрації, він є своєрідним мостом між WordPress і Magento, що забезпечує вирішення конфліктів між даними системами. Також, даний клас зберігає у собі найважливіші об'єкти Magento, за допомогою яких можливе по-дальше розширення функціоналу інтеґрації.

При реалізації системи інтеґрації було найбільш точно відпрацьовано етапи ініціалізації та повного запуску Magento в середовищі WordPress, що дозволило в подальшому масштабувати систему інтеґрації.

Спершу створюється об'єкт для збору помилок, адже ініціалізація, як правило, робиться лише один раз, а при наступній ініціалізації об'єкт для збору помилок має бути новим, щоб було можливо проаналізувати різницю. Далі, перевіряється функція перекладу в системі WordPress. Якщо клієнт замінив її на варіант, запропонований документацією плаґіну інтеґрації, то дана функція вважається оптимізованою для використання із системою Magento. Інакше, генерується попередження, що функція перекладу неоптимізована, а ініціалізація завершується. Після успішної перевірки на функцію перекладу ініціалізується коренева директорія Magento: перевіряється правильність шляху, за умови AJAX виклику зі сторінки базових налаштувань береться директорія, передана із даними POST запиту. Далі, ініціалізується Bootstrap об'єкт – один із головних об'єктів Magento, без якого неможливий жоден запуск системи, окрім того він створює ObjectManager – об'єкт, що дозволяє створювати екземпляри будь-яких класів Magento. В свою чергу, ObjectManager є сховищем екземплярів вже створених об'єктів, а тому, він реалізовує твірний шаблон [7] проєктування «Object pool» і є дуже важливим для подальшої роботи Magento системи. При ініціалізації Bootstrap об'єкту також створюється об'єкт автозавантажувача усіх класів Magento. Якщо Bootstrap об'єкт було ініціалізовано успішно, то його екземпляр буде збережено. У випадку відсутності даного об'єкту процес ініціалізації Magento завершується. За наявності даного об'єкту ініціалізація продовжується. Виконується модифікація середовища веб-серверу методом оптимізації суперглобальної змінної $_SERVER для безконфліктної роботи Magento, а попередній стан $_SERVER зберігається. Далі, ініціалізується об'єкт App (повна назва \Magento\Framework\App\Http) через виклик «createApplication(…)» методу Bootstrap об'єкту. Відновлюється попередньо збережений стан $_SERVER. Наприкінці встановлюється інформація про можливе здійснення повного запуску Magento записом булевого значення «true» в поле класу M2I_External з назвою «can_launch».

В алгоритмі повного запуску Magento варто відзначити деякі мо-менти. Перевірка на те, чи був вже здійснений запуск, дозволяє зберегти час виконання завантаження системи WordPress. Очищення full_page кешу є дієвим лише для Magento нижче 2.2.5 версії і повинно бути видалене в наступних версіях плаґіну. Збереження об'єкту Layout в ядрі потрібне для майбутнього отримання блоків і контейнерів при інтеґрації вмісту Magento у WordPress. Збереження об'єкту крамниці (Store) потрібне для передачі правильних посилань та інших можливих перевірок. Встановлення потреби перекладу зі сторони Magento потріб-не для ліквідації невідповідності параметрів між функцією перекладу WordPress та функцією перекладу Magento.

Таким чином, було розглянуто алгоритми ініціалізації та повного запуску інтеґрації Magento 2 у WordPress та основні класи системи. Розроблений плаґін використовується для забезпечення електронних магазинів на Magento 2 сучасним та багатофункціональним блоґом на СКВ «WordPress».