O'REILLY®

2-е издание

Разработка веб-приложений на WordPress

WordPress как фреймворк

Брайан Мессенленер
Джейсон Коулман

bhv®

SECOND EDITION

Building Web Apps with WordPress

WordPress as an Application Framework

Brian Messenlehner and Jason Coleman

Beijing • Boston • Farnham • Sebastopol • Tokyo O'REILLY®

Брайан Мессенленер
Джейсон Коулман

Разработка веб-приложений на WordPress

WordPress как фреймворк

Санкт-Петербург
«БХВ-Петербург»
2021

УДК 004.4'236
ББК 32.973.26-018
М53

Мессенленер, Б.

М53 Разработка веб-приложений на WordPress: Пер. с англ. / Б. Мессенленер, Д. Коулман. — 2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2021. — 528 с.: ил.

ISBN 978-5-9775-6753-4

Подробно рассматривается создание веб-приложений на платформе WordPress, в том числе для мобильных устройств, принципы работы таких приложений. Описана структура каталогов и базы данных, приведены типы записей, метаданных и таксономий, перечислены основные классы и функции. Уделено внимание разработке собственной темы оформления с адаптивным дизайном. Рассказывается о типах пользователей и их ролях в архитектуре WordPress. Отдельная глава посвящена работе с API-интерфейсами, объектами и вспомогательными функциями, рассматриваются вопросы безопасности веб-приложений, принципы написания безопасного кода. Изучается REST API в WordPress, JavaScript-фреймворки, способы локализации приложений. Описаны принципы построения многосайтовых сетей, оптимизации и масштабирования. Рассматривается проект Gutenberg и его возможности.

Во втором издании авторы рассматривают новые функции и возможности актуальных версий WordPress. Все примеры кода из книги доступны на веб-сервисе GitHub.

Для веб-разработчиков

УДК 004.4'236
ББК 32.973.26-018

Группа подготовки издания:

Руководитель проекта	*Павел Шалин*
Зав. редакцией	*Людмила Гауль*
Перевод с английского	*Михаила Райтмана*
Компьютерная верстка	*Натальи Смирновой*
Оформление обложки	*Карины Соловьевой*

"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

ISBN 978-1-491-99008-7 (англ.)
ISBN 978-5-9775-6753-4 (рус.)

Оглавление

Вступительное слово

В течение последних нескольких лет я проводил от 15 до 24 выходных каждый год, встречаясь с разработчиками программного обеспечения по всей стране. В итоге после примерно 25 тыс. диалогов я убедился в трех простых фактах:

- Веб-приложения встречаются повсеместно.

  Трудно представить жизнь без веб-приложений. Сегодня мы управляем нашими банковскими счетами онлайн. Мы бронируем отели и отдых онлайн с помощью веб-приложений; мы платим наши налоги онлайн в чем-то вроде приложения; мы записываемся в парикмахерскую перед визитом, используя то, что кажется простой формой, но это тоже приложение. Веб-приложения вездесущи.

- Веб-разработчики чувствуют себя совершенно неспособными создавать веб-приложения.

  Если вы поговорите с обычными пользователями, то каждый из них скажет, что у него есть друг-разработчик — гений, который может делать что угодно с компьютером. Но если вы потратите время на разговоры непосредственно с разработчиками, то обнаружите, что они видят мир совсем по-другому.

  Они смотрят на веб-приложения, которые считают "реальными", и на компании, стоящие за ними, и убеждены, что люди, которые работают "там", знают о создании веб-приложений то, что никому не известно.

  Разработчики убеждены, что в одиночку могут создавать только небольшие приложения, и они не знают, что же им нужно для создания такого приложения, которое станет невероятно популярным.

- Чего не хватает, так это немного информации и гораздо больше уверенности.

  Когда вы разговариваете с разработчиками о веб-приложениях, то быстро обнаруживаете, что отсутствующий элемент — это уверенность, ее не хватает больше всего. Написать программный код не сложно, если вы знаете, что необходимо делать. Но это не значит, что разработчикам не нужно понимать процесс глубже.

Наши беседы проходили по всей стране на местных конференциях под названием WordCamps. Это общественные мероприятия, которые обычно посещают от 200 до 500 участников — в равной степени как разработчики, так и пользователи приложений. Но все они используют (или намереваются это сделать) продукт веб-публикации под названием WordPress.

На WordPress, если вы никогда не слышали о нем, работает треть интернет-сайтов на планете. Он в основном служит в качестве платформы для публикации контента. Но на самом деле с помощью WordPress можно также запустить ваше веб-приложение.

Об этом пойдет речь на страницах данной книги.

Мне нравится в этой книге, что первые главы достаточно подробны. Это помогает читателям медленно погрузиться в предмет и придает уверенность. И я знаю, насколько важным это обстоятельство будет для вас. Образно выражаясь, книга создает "дорожную карту", показывая, как добраться от того места, где вы находитесь, до того места, где вы хотите оказаться.

Но ценность этой книги заключается еще и в том, что она глубже "вдавливает" вас в детали каждой строки кода, которую вам нужно написать. Что-то будет новым для вас, что-то — нет. Но чего вы здесь точно не найдете, так это абстрактных описаний, которые никак не объясняются и оставляют открытым вопрос, как применить полученные знания.

Брайан и Джейсон знакомят вас с каждой составной частью WordPress, которую вы должны знать, чтобы применить его в качестве основы веб-приложения. Сегодня существуют курсы программирования, ориентированные на преподавание именно этого материала. К счастью, вам не придется платить за подобные курсы, которые стоят намного дороже, чем эта книга. И это, пожалуй, лучшая реклама данной книги.

До сих пор вы, возможно, думали, что есть два типа разработчиков программ. Первые — те, кто создает сложные аппаратно-ориентированные приложения, которые вам не под силу. Вторые, как и вы, разрабатывают небольшие простые приложения.

Когда вы прочитаете эту книгу до конца, изучите теорию и "поиграете" с программным кодом, когда вы заставите его работать, то поймете, что существует только один вид разработчика: тот, который способен создавать все, что только можно представить, и вы станете именно таким разработчиком.

Тот факт, что вы сейчас читаете эти строки, уже подтверждает сказанное. Теперь вам пора идти дальше и реализовать свой потенциал.

Крис Лема, вице-президент по продуктам и инновациям в Liquid Web

Предисловие

На момент написания этой книги WordPress поддерживает 32% всех сайтов в Интернете, и это число растет. Многие разработчики хотят расширить возможности своих сайтов WordPress, но считают, что им нужно перейти к более традиционной среде приложений, такой как Ruby on Rails, Symfony, Yii или Laravel, для создания "настоящих" веб-приложений. Такое мнение неверно, и мы здесь, чтобы исправить это заблуждение.

Несмотря на то, что WordPress изначально был программным обеспечением для ведения блогов и в настоящее время существует в основном как система управления контентом, постепенно он превратился в гибкую и функциональную платформу для создания веб-приложений. Эта книга покажет вам, как использовать WordPress в качестве фреймворка для создания *любого* веб-приложения, большого или маленького.

Для кого предназначена эта книга

Эта книга будет наиболее полезна для разработчиков WordPress, желающих создавать более "тяжелые" приложения, и для программистов PHP, имеющих некоторый опыт работы с WordPress, ищущих среду создания приложений на основе PHP.

Коммерческие разработчики плагинов и те, кто работает над крупными распределенными проектами WordPress, также найдут полезными концепции и методы этой книги.

Если вы PHP-программист или независимый от языка разработчик, использующий другую платформу, и пользуетесь огромной библиотекой плагинов и тем WordPress, то вы возможно удивитесь, узнав, насколько хорошо WordPress может работать как фреймворк. Чтение этой книги и выполнение упражнений из нее могут изменить вашу трудовую жизнь в лучшую сторону.

Мы предполагаем, что читатели имеют общие сведения о программировании на PHP. Вы также должны обладать базовыми знаниями HTML и CSS, а также быть знакомыми с MySQL и SQL-запросами. Понимание основ JavaScript и программирования jQuery понадобится в *главе 9* и при рассмотрении связанных примеров.

Для кого НЕ предназначена эта книга

Эта книга не для тех, кто хочет научиться работать с WordPress в качестве пользователя. В ней изложены краткие введения о стандартной функциональности WordPress, но мы предполагаем, что вы уже знакомы с WordPress с точки зрения пользователя.

Эта книга не предназначена для непрограммистов. Несмотря на то, что можно создавать очень функциональные веб-приложения, просто комбинируя и конфигурируя множество плагинов, доступных для WordPress, эта книга написана для разработчиков, создающих свои собственные плагины и темы для новых веб-приложений.

Эта книга не научит вас программировать, а скорее научит программировать "путем WordPress".

Структура книги

Мы надеемся, что благодаря этой книге вы изучите методы программирования и проектирования, а также освоите лучшие практики для разработки сложных приложений с использованием WordPress.

Глава 1 определяет, что мы подразумеваем под "веб-приложением", а также описывает, почему следует (или не следует) создавать веб-приложения на WordPress, и как сравнивать WordPress с другими фреймворками. Мы также представляем SchoolPress, приложение WordPress, которое служит нам в качестве примера на протяжении всей книги.

Глава 2 охватывает основы WordPress. Мы просматриваем различные каталоги базовой установки WordPress. Мы также объясняем каждую таблицу базы данных, созданную WordPress, какие данные хранятся в каждой и какие функции WordPress отображаются на эти таблицы. Даже если вы опытный разработчик WordPress, вы можете кое-что узнать из этой главы, и мы рекомендуем вам прочитать ее.

Глава 3 повествует все о плагинах. Что это такое? Как создавать свои собственные плагины? Как вы должны структурировать основной плагин вашего приложения? Когда вы должны использовать сторонние плагины или свои собственные?

Глава 4 — это все о темах. Как работают темы? Как темы отображаются на представления в типичной структуре "модель – представление – контроллер" (MVC)? Какой код должен входить в вашу тему, а какой — в плагины? Еще мы рассмотрим использование фреймворков тем и UI, а также познакомимся с основами адаптивного дизайна.

Глава 5 описывает пользовательские типы записей и таксономию. Мы рассмотрим стандартные типы записей, встроенные в WordPress, выясним, почему вам может потребоваться создать свои собственные, а затем опишем, как это сделать. Мы также расскажем о метаданных и таксономиях публикаций, о том, для чего каждый из них подходит, и о том, как создавать собственные таксономии и сопоставлять их с

вашими типами записей. Наконец, мы покажем, как создавать классы-обертки для ваших типов записей, чтобы оптимизировать ваш код с помощью объектно-ориентированного программирования (ООП).

Глава 6 посвящена пользователям, их роли и возможностям. Мы покажем, как добавлять, обновлять и удалять пользователей программно и как работать с метаролями, ролями и правами пользователей. Мы также покажем, как расширить класс `WP_User` для ваших пользовательских архетипов, таких как "клиенты" и "учителя", чтобы лучше организовать ваш код на основе методов ООП.

Глава 7 охватывает некоторые из более полезных API-интерфейсов WordPress и вспомогательных функций, которые не вписываются в остальную часть книги, но все еще важны для разработчиков, создающих веб-приложения с помощью WordPress.

Глава 8 содержит все сведения о защите ваших приложений WordPress, плагинов и тем.

Глава 9 охватывает использование JavaScript и Ajax в вашем приложении WordPress. Мы расскажем о правильном способе включения JavaScript в WordPress и о том, как создать асинхронное поведение в вашем приложении.

Глава 10 рассказывает о REST API для WordPress и о том, как применить его для интеграции WordPress с внешними приложениями.

Глава 11 посвящена описанию редактора блоков и созданию своих собственных блоков.

Глава 12 рассматривает многосайтовые сети WordPress, включая их настройку и обстоятельства, которые следует учитывать при разработке для сети.

Глава 13 описывает локализацию ваших плагинов и тем для WordPress, в том числе вопрос, как подготовить код для перевода и как создавать и использовать файлы перевода.

Глава 14 раскрывает, как оптимизировать и масштабировать WordPress для больших веб-приложений. Мы рассмотрим, как проверить производительность вашего приложения WordPress и укажем наиболее популярные методы ускорения и масштабирования сайтов под управлением WordPress.

Глава 15 посвящена созданию приложения для интернет-торговли. Мы рассмотрим различные типы доступных плагинов для электронной коммерции и способы выбора между ними. Затем мы подробно опишем, как с помощью WordPress можно обрабатывать платежи и управлять учетными записями веб-приложения SaaS (software as a service).

Глава 16 раскрывает, как использовать WordPress для запуска собственных приложений на мобильных устройствах, создавая оболочки приложений для iOS и Android.

Глава 17 описывает некоторые сторонние библиотеки PHP, службы и API, которые часто встречаются в веб-приложениях, а также способы их интеграции с WordPress, включая полную миграцию.

Глава 18 предсказывает будущее WordPress, описывает, какие приложения мы ожидаем увидеть на WordPress, какие обновления мы ожидаем для WordPress, а также на какие инструменты и платформы следует обратить внимание в будущем.

О коде программ

Все примеры из этой книги вы можете найти по адресу **github.com/bwawwp**. Обратите внимание, что эти примеры кода были написаны для того, чтобы наиболее четко передать концепции, которые мы рассмотрим в книге. Чтобы улучшить удобочитаемость листингов программ, мы часто игнорируем лучшие практики стандартов программирования (**oreil.ly/xw3dV**), безопасность и локализацию (которые мы рассмотрим в *главах 8* и *13*) или не рассматриваем некоторые особые случаи. Вы должны иметь это в виду, прежде чем использовать какие-либо примеры в своем рабочем коде.

Образец приложения SchoolPress доступен по адресу **schoolpress.me**, с открытым исходным кодом для этого сайта по адресу **oreil.ly/6Lbax**.

Условные обозначения

В этой книге приняты следующие обозначения:

Курсивный шрифт — указывает на новые термины, имена файлов и расширения файлов.

Полужирный шрифт — им оформлены URL-адреса и адреса электронной почты.

`Моноширинный шрифт` — используется для листингов программ, а также в абзацах для ссылки на элементы программы, такие как имена переменных или функций, базы данных, типы данных, переменные среды, операторы и ключевые слова.

`Полужирный моноширинный шрифт` — указывает на команды или другой текст, который должен быть набран пользователем.

`Курсивный моноширинный шрифт` — показывает текст, который следует заменить предоставленными пользователем значениями или значениями, определенными контекстом.

Данный элемент обозначает подсказку или совет.

Данный элемент обозначает примечание или замечание.

Данный элемент обозначает предупреждение или предостережение.

Использование примеров программного кода

Как было указано в разделе "О коде программ", все примеры программного кода в этой книге можно найти по адресу **github.com/bwawwp**.

Если у вас появится технический вопрос или проблема с использованием примеров кода, отправьте электронное письмо по адресу **bookquestions@oreilly.com**.

Эта книга призвана помочь вам выполнить свою работу. В общем случае, если пример кода приведен в книге, то вы можете вставить его в свои программы и документацию. Вам не нужно обращаться к нам за разрешением, если вы не воспроизводите значительную часть листинга программы. Например, написание программы, которая содержит несколько фрагментов кода из этой книги, не требует разрешения. Для продажи или распространения примеров из книг O'Reilly потребуется разрешение. Чтобы ответить на какой-либо вопрос, сославшись на эту книгу и приведя пример кода, разрешения не нужно. Включение значительного объема кода программ из этой книги в документацию вашего продукта требует разрешения.

Для нас важно, если при ссылке вы укажете атрибуты издания, но мы этого не требуем. Атрибуты обычно включают название книги, автора, издателя и ISBN. Например: "*Создание веб-приложений с WordPress*, второе издание, Брайан Мессенленер и Джейсон Коулман (O'Reilly). Все права принадлежат Брайану Мессенленеру и Джейсону Коулману, 978-1-491-99008-7 2020."

Если вы считаете, что использование примеров кода требует разрешения, свяжитесь с нами по адресу **permissions@oreilly.com**.

Благодарности

От Брайана: спасибо Джейсону Коулману и Мэтту Мулленвегу, я не смог бы написать эту книгу без их помощи! Особая благодарность Алисии Янг за то, что она остается в курсе событий в O'Reilly Media. Выражаю также благодарность нашим техническим экспертам за то, что все в книге является верным. Спасибо Скотту Болинжеру из AppPresser.com и Джеффу Уорлею из AlphaWeb.com за то, что терпели меня. Привет семье и друзьям, которые всегда были рядом со мной и никогда не переставали верить в меня. Больше всего я благодарен своим детям: Дали, Брайану-младшему, Нине, Кэму и Акселю Мессенленерам — они дают мне цель, и без них я, вероятно, даже не знал бы, что такое WordPress.

От Джейсона: спасибо моему соавтору Брайану за то, что он попросил меня написать эту книгу вместе с ним. Спасибо нашим первым редакторам Меган и Аллисон за то, что они следили за нами и помогали нам оставаться верными нашему первоначальному видению. Спасибо Алисии Янг за редактирование второго издания этой книги и за всю проделанную работу с нашими объяснениями "WP Drama". Выражаю благодарность всем замечательным техническим редакторам, которые были у нас в обеих редакциях книги: Сэм Хотчкисс, Питер Макинтайр, Пиппин Уильямсон, Джон Джеймс Джейкоби и Эндрю Лима. Спасибо Фредерику Таунсу за его отзывы и вклад в нашу главу по оптимизации и масштабированию. Спасибо Крису Леме за замечательный перевод этой книги, его отзывы о книге и советы в целом. Благодарю всех в сообществе WordPress, кто ответил на все мои случайные твиты и, возможно, знал или не знал, что они помогает мне написать эту книгу. Спасибо моей жене, Ким, за то, что она поддержала меня, как всегда, во время еще одного приключения в нашей жизни. Спасибо моей дочери Марин за то, что она скучала по мне, когда я отсутствовал, чтобы писать, и моему сыну, Исааку, за то, что постоянно спрашивал меня, "закончил ли я уже книгу". И последнее, но не менее важное — спасибо моей семье, которая всегда поддерживала меня во время написания книги: мама, папа, Джереми и Нана Мэн с нетерпением ждут, когда станут первыми непрограммистами, прочитавшими эту книгу.

Создание веб-приложений с помощью WordPress

Эта книга поможет вам создать что угодно с помощью WordPress: веб-сайты, темы, плагины, веб-сервисы и веб-приложения. Мы решили сосредоточиться на веб-приложениях, потому что вы можете рассматривать их как супер-сайты, использующие все методы, которые мы опишем.

Многие считают, что WordPress недостаточно мощен или не предназначен для создания веб-приложений; далее мы еще вернемся к этому вопросу. Мы много лет создавали веб-приложения на WordPress и знаем, что с его помощью вы можете создавать масштабируемые приложения.

В этой главе мы начнем с определения того, что такое веб-приложение, а затем выясним, почему WordPress является отличным фреймворком для их создания. Мы также опишем некоторые ситуации, в которых применение WordPress *не будет* лучшим способом для создания вашего веб-приложения.

Что такое веб-сайт?

Вы знаете, что такое веб-сайт: набор из одной или нескольких веб-страниц, содержащих информацию, доступ к которой осуществляется через веб-браузер.

Что такое приложение?

Нам нравится определение в Википедии (**oreil.ly/2DUKl**): "Прикладное программное обеспечение (сокращенно приложение) — это программное обеспечение, предназначенное для выполнения группы скоординированных функций, задач или действий на благо пользователя".

Что такое веб-приложение?

Веб-приложение — это всего лишь приложение, запускаемое через веб-браузер.

Обратите внимание, что в некоторых веб-приложениях технология браузера скрыта, например, когда вы интегрируете веб-приложение в собственное приложение для Android или iOS, запускаете веб-сайт как приложение в Google Chrome или активизируете приложение с помощью Adobe AIR. Однако внутри этих приложений по-прежнему существует система парсинга HTML, CSS и JavaScript.

Вы также можете думать о веб-приложении как о *веб-сайте с некоторым дополнительным функционалом приложения*. Не существует точной разделительной линии, за которой веб-сайт становится веб-приложением. Это один из тех случаев, когда данное обстоятельство совершенно очевидно для вас.

Мы *можем* лишь объяснить некоторые функции веб-приложения, дать вам несколько примеров, а затем попытаться придумать сокращенное определение, чтобы вы в целом понимали, о чем мы говорим, когда употребляем этот термин в книге.

При чтении этой книги вам встретятся ссылки на SchoolPress. SchoolPress — это веб-приложение, которое мы создаем, чтобы помочь школам и преподавателям управлять своими учениками и учебными программами. Все примеры кода направлены на функциональность, которая может существовать в SchoolPress. Более подробно об общей концепции SchoolPress мы поговорим далее в этой главе.

Функции веб-приложения

Далее перечислены некоторые функции, обычно связанные с веб-приложениями и приложениями в целом. Чем больше этих функций есть на веб-сайте, тем целесообразнее считать его веб-приложением[1].

- *Интерактивные элементы*

  Типичный веб-сайт включает в себя навигацию по страницам, прокрутку и нажатие гиперссылок. Веб-приложения также могут иметь ссылки и прокрутку, но чаще встречаются другие методы навигации по приложению.

  Веб-сайты с формами предлагают функционал совершения транзакций. Примером может служить форма обратной связи на веб-сайте или форма заявки на странице вакансий на веб-сайте компании. Формы позволяют пользователям взаимодействовать с сайтом, используя нечто большее, чем клик.

  Веб-приложения будут иметь еще больше элементов интерактивного пользовательского интерфейса (UI — user interface). Примеры включают в себя панели инструментов, элементы перетаскивания, редакторы форматированного текста и ползунки.

- *Задачи, а не содержание*

  Помните, что веб-приложения "разработаны, чтобы помочь пользователю выполнять определенные задачи". Пользователи Google Maps получают маршруты проезда. Пользователи Gmail пишут электронные письма. Пользователи Trello управляют списками. Пользователи SchoolPress добавляют комментарии по теме урока.

  Некоторые приложения по-прежнему ориентированы на контент. Типичный сеанс Facebook или Twitter почти на 90% состоит из чтения. Тем не менее сами

[1] На многие идеи в этом разделе повлияли сообщения в блоге: "Что такое веб-приложение?" Доминика Хазаэль-Массо (**bit.ly/wiawa**) и Боба Бэксли (**bit.ly/wiawa2**).

приложения представляют способ просмотра контента, отличный от обычного просмотра веб-страниц.

- *Логины*

  Вход в систему и учетные записи позволяют веб-приложению сохранять информацию о своих пользователях. Эта информация предназначена для облегчения основных задач приложения и обеспечения постоянного взаимодействия. При входе в систему пользователи SchoolPress могут видеть, какие сообщения являются непрочитанными. У них также есть имя пользователя, которое идентифицирует их деятельность в приложении.

  Веб-приложения также могут иметь уровни пользователей. В SchoolPress будут администраторы, контролирующие внутреннюю работу приложения, учителя, управляющие процессом обучения и ученики, участвующие в дискуссиях в классе.

- *Возможности устройства*

  Веб-приложения, работающие на вашем телефоне, могут получить доступ к вашей камере, адресной книге, внутреннему хранилищу и информации о местоположении GPS. Веб-приложения, функционирующие на персональном компьютере, могут иметь доступ к веб-камере или локальному жесткому диску. Одно и то же веб-приложение может реагировать по-разному в зависимости от устройства, на котором оно запущено. Веб-приложения будут настраиваться для разных размеров экранов, их разрешений и возможностей.

- *Работа в автономном режиме*

  Всегда, когда это возможно, желательно, чтобы ваши веб-приложения работали в автономном режиме. Конечно, интерактивность Интернета главным образом определяет "веб" часть веб-приложения, но сайт, который все еще работает, когда вы проезжаете через туннель, будет больше походить на приложение.

  С Gmail вы можете создавать черновики писем в автономном режиме. Evernote позволяет создавать заметки офлайн, а затем синхронизировать их с Интернетом после восстановления подключения.

- *Мэшапы*

  Веб-приложения могут связывать одно или несколько веб-приложений вместе. Веб-приложение может использовать различные веб-сервисы и API для передачи и извлечения данных. У вас может быть веб-приложение, которое получает информацию о местоположении, такую как долгота и широта, из Twitter и отправлять и отмечать его на карте Google Map.

Мобильные приложения

С тех пор, как первое издание этой книги было опубликовано еще в 2012 году, веб-приложения, в частности мобильные приложения, получили широкое распространение. На большинстве веб-сайтов мобильные устройства уже преобладают

над ПК и являются крупнейшим источником трафика (Источник: Perficient, Inc. (**oreil.ly/N92kX**)).

В 2012 году типичное веб-приложение выглядело как Basecamp — менеджер проектов, доступ к которому осуществлялся через веб-браузер на вашем компьютере. В 2019 году типичное веб-приложение выглядит как Twitter — приложение для коммуникации, доступ к которому можно получить с помощью телефона iOS или Android.

Поскольку в большинстве случаев многие из пользователей будут получать доступ к вашим веб-сайтам и приложениям на мобильном устройстве, при разработке веб-приложений мы придерживаемся взгляда "прежде всего для мобильных устройств". О том, как заставить ваши приложения WordPress работать на мобильных устройствах, речь пойдет в *главе 16*. А об основах адаптивного дизайна и о том, как заставить ваши веб-сайты правильно отображаться на экранах любого размера мы расскажем в *главе 4*.

Прогрессивные веб-приложения

Прогрессивные веб-приложения (PWA) — это веб-сайты, реализующие преимущества современных функций браузера, которые ведут себя как собственные приложения для Android, iOS или ПК. В частности, веб-сайты, которые используют *сервис-воркеры* (service workers) для работы в автономном режиме, содержат файл манифеста веб-приложения для определения приложения в операционной системе (ОС) и отвечают нескольким другим требованиям, поэтому могут быть запущены как приложения прямо из браузера.

Идея PWA была разработана командой Google Chrome, но теперь поддерживается на iOS и в большинстве современных веб-браузеров. Плагин для поддержки PWA (**oreil.ly/gThAQ**) разрабатывается для работы основных функций PWA в WordPress core. С помощью этого плагина вы можете превратить ваш сайт WordPress в PWA, и это хорошая идея. Но на самом деле создание PWA — это скорее образ мышления, а не простой переход. Аналогично "функциям веб-приложения", которые мы только что описали, главный сайт PWA Google (**oreil.ly/FAwTK**) содержит контрольный список функций, необходимых в большинстве PWA (**oreil.ly/HBuTg**). Укажем следующие базовые функции:

- Сайт обслуживается по HTTPS.
- Страницы адаптированы для планшетов и мобильных устройств.
- Все URL-адреса приложения загружаются в автономном режиме.
- Метаданные предоставляются для добавления на главный экран.
- Первая загрузка быстрая даже с 3G.
- Сайт работает независимо от браузера.
- Переход между страницами не ощущается как работа с сетью.
- У каждой страницы есть URL.

В дополнение к базовым функциям есть контрольный список элементов для "примерных" PWA, который охватывает пользовательский опыт (UX) и производительность. Инструмент Google Lighthouse (**oreil.ly/GwEkb**) предоставляет автоматизированные тесты и отчеты для соответствия критериям PWA. Даже разработчики полностью нативных приложений или приложений для браузера, могут воспользоваться некоторыми советами из контрольных списков PWA и отчетов Lighthouse.

Зачем нужен WordPress?

Ни один язык программирования или программный инструмент не подойдет для любой разработки. Мы еще коснемся вопроса, почему вы *не* захотите использовать WordPress, но сейчас давайте рассмотрим некоторые ситуации, в которых создавать веб-приложения целесообразно именно с помощью WordPress.

Вы уже используете WordPress

Если вы уже используете WordPress для своего основного сайта, то можете просто добавить плагин, который вам необходим. В WordPress есть отличные плагины для электронной торговли (WooCommerce), форумов (bbPress), сайтов с подпиской (Paid Memberships Pro), функций социальных сетей (BuddyPress) и геймификации (BadgeOS).

Встраивание приложения в существующий сайт WordPress сэкономит ваше время и упростит работу для ваших пользователей. Итак, если ваше приложение достаточно простое, вы можете создать собственный плагин на своем сайте WordPress для программирования функциональности вашего веб-приложения.

Когда вы довольны своим сайтом на WordPress, не поддавайтесь искушению, если люди говорят, что вам нужно перейти на что-то другое, чтобы добавить определенные функции на ваш сайт. Это, скорее всего, неправда. Вам не нужно выбрасывать всю работу, которую вы уже проделали в WordPress, и последующие доводы — это веские причины придерживаться WordPress.

С помощью WordPress легко управлять контентом

Разработанный сначала как платформа для ведения блогов, с введением пользовательских типов записей (англ. CPT — Custom Post Type) в версии 3.0 WordPress развился в полностью функциональную систему управления контентом (англ. CMS — Content Management System). Любая страница или сообщение может быть отредактировано администратором через панель инструментов, доступ к которой можно получить через ваш веб-браузер. О работе с CPT вы узнаете в *главе 5*.

WordPress упрощает добавление и редактирование контента с помощью редактора WYSIWYG (What You See Is What You Get — "Что видишь, то и получаешь"), поэтому вам не нужно привлекать веб-дизайнеров каждый раз, когда вы хотите внести простые изменения в свой сайт. Вы также можете создавать собственные меню и элементы навигации для своего сайта, не касаясь программного кода.

Если ваше веб-приложение сфокусировано на фрагментах контента (например, наше приложение SchoolPress ориентировано на назначение пользователям заданий и их обсуждение), API пользовательских типов постов для WordPress (описанный в *главе 5*) позволяет легко настроить этот пользовательский контент и управлять им.

Даже приложения, более ориентированные на задачи, обычно имеют несколько страниц с информацией, документацией и продажами. Использование WordPress для вашего приложения даст вам возможность управления приложением и всем вашим контентом из одного места.

WordPress позволяет просто и безопасно управлять пользователями

В WordPress есть все необходимое для добавления на сайт как администраторов, так и конечных пользователей.

Помимо управления доступом к контенту, система ролей и возможностей в WordPress расширяема и позволяет вам контролировать, какие *действия* доступны для определенных групп пользователей. Например, по умолчанию пользователи с ролью участника могут *добавлять* новые сообщения, но не могут *публиковать* их. Точно так же вы можете создавать новые роли и возможности для управления доступом к вашим пользовательским функциям.

Вы можете использовать плагины, такие как Paid Memberships Pro, чтобы расширить встроенное управление пользователями и назначать членов разных уровней и контролировать доступ пользователей к контенту. Например, вы можете создать уровень, чтобы предоставить участникам с платным аккаунтом доступ к премиум-контенту на вашем сайте WordPress.

Плагины

В репозитории WordPress (**wordpress.org/plugins/**) имеется более 55 тыс. бесплатных плагинов. Существует множество дополнительных плагинов, как бесплатных, так и коммерческих, на различных сайтах в Интернете. Если у вас появляется идея для расширения вашего сайта, велика вероятность, что для этого уже есть плагин, который сэкономит ваше время и деньги.

Существует несколько необходимых плагинов, которые мы задействуем практически на каждом создаваемом нами сайте и веб-приложении.

Для большинства разрабатываемых веб-сайтов вы, наверное, хотите кэшировать вывод для ускорения просмотра, использовать такие инструменты, как Google Analytics для отслеживания посетителей, создавать карты сайта и настраивать набор страниц для поисковой оптимизации (англ. SEO — Search Engine Optimization), а также выполнять ряд других общих задач.

Есть много хорошо поддерживаемых плагинов для всех этих функций. Мы предлагаем наши любимые в этой книге. Вы можете найти их список на веб-сайте этой книги (**bwawwp.com/plugins/**).

Гибкость важна

WordPress — полноценный фреймворк, способный на многое. Кроме того, WordPress построен на основе технологий PHP, JavaScript и MySQL, поэтому все, что вы можете встроить в PHP/MySQL (а это почти все), может быть достаточно легко встроено в ваше приложение WordPress.

WordPress и PHP/MySQL в целом не идеально подходят для любой задачи, но они пригодны для широкого круга задач. Наличие одной платформы, которая будет расти вместе с вашим бизнесом, позволит вам быстрее выполнять задачи и перестраиваться. Например, вот типичный пример сайта запуска стартапа Lean, работающего на WordPress:

- Объявите о своем стартапе с помощью одностраничного сайта.
- Добавьте форму для сбора адресов электронной почты.
- Добавить блог.
- Сосредоточьтесь на SEO и оптимизируйте весь контент.
- Отправляйте посты блога в Twitter и Facebook.
- Добавьте форумы.
- Используйте плагин Paid Memberships Pro, чтобы позволить участникам платить за доступ.
- Добавьте пользовательские формы, инструменты и поведение приложения для участников с платной подпиской.
- Обновите пользовательский интерфейс, используя JavaScript и фреймворки.
- Настройте сайт и сервер для масштабирования.
- Локализуйте сайт/приложение для разных стран и языков.
- Добавьте поддержку Progressive Web App.
- Запустите обертки для iOS и Android для приложения.

Суть, почему нужно идти по этому пути, состоит в том, что на каждом этапе у вас есть *одна и та же база данных пользователей* и *одна и та же платформа разработки*.

Частые обновления безопасности

Тот факт, что WordPress используется на миллионах сайтов, делает его целью хакеров, пытающихся найти уязвимости в его безопасности. Некоторые из этих хакеров были успешны в прошлом, однако разработчики WordPress быстро отслеживают уязвимости и выпускают обновления для их устранения. Получается, будто миллионы людей постоянно тестируют и исправляют ваше программное обеспечение.

Базовая архитектура WordPress делает применение этих обновлений быстрым и безболезненным процессом, который могут выполнять даже начинающие веб-

пользователи. Если вы хорошо знаете, как настроить WordPress и обновить его до последних версий, когда они станут доступны, WordPress станет гораздо более безопасной платформой для вашего сайта, чем все остальное. Мы обсудим безопасность более подробно в *главе 8*.

Стоимость

WordPress — бесплатный продукт. PHP бесплатен. MySQL бесплатен. Большинство плагинов тоже бесплатны.

Серверы и хостинг стоят денег, но в зависимости от того, насколько велико ваше веб-приложение и сколько трафика вы получаете, это может быть относительно недорого. Если вам требуются пользовательские функции, которых нет ни в одном из существующих плагинов, вам, возможно, придется заплатить разработчику за создание нового. Или, если вы сами разработчик, это будет стоить вам времени.

Ответы на некоторые распространенные критические мнения о WordPress

Некоторые высококвалифицированные критики WordPress могут сказать, что это не очень хорошая основа для создания веб-приложений и это вообще не фреймворк. При всем уважении к тем, кто придерживается подобных мнений, нужно объяснить, почему мы не согласны. Далее приведены некоторые распространенные критические замечания.

WordPress подходит только для блогов

Многие люди считают, что поскольку WordPress впервые был создан для ведения блога, он хорош только для этой цели.

Подобные заявления были верны несколько лет назад, но с тех пор WordPress внедрил мощную функциональность CMS, что делает его полезным для других сайтов, ориентированных на контент. WordPress в настоящее время является самой популярной из используемых CMS с долей рынка более 60%[2]. На рис. 1.1 показан слайд из презентации Мэтта Мулленвега "Состояние WordPress" с конференции WordCamp San Francisco 2013. Перевернутая пирамида слева изображает WordPress 2006 года, где бóльшая часть кода посвящена приложению блога, и есть немного кода CMS и платформы, на котором она держится. Правая пирамида представляет текущее состояние платформы WordPress, где бóльшая часть кода находится в самой платформе, поверх которой расположен слой CMS, а поверх уровня CMS — приложение для блога. Сейчас WordPress является гораздо более устойчивой платформой, чем несколько лет назад.

[2] W3Tech (**bit.ly/w3techs**) регулярно проводит исследования по использованию различных систем управления контентом.

Рис. 1.1. Диаграммы из презентации Мэтта Мулленвега "Состояние WordPress" 2013 год. WordPress не всегда был таким устойчивым

С помощью API пользовательских типов записей можно настроить установку WordPress для поддержки других типов контента, кроме постов или страниц блога. Мы подробно рассмотрим это в *главе 5*.

WordPress предназначен только сайтов, ориентированных на контент

Подобно людям с позицией "только для блогов", некоторые скажут, что WordPress предназначен только для управления контентом сайтов.

Во-первых, даже если бы WordPress был применим только к контентным сайтам и приложениям, на него приходилось бы большое количество приложений. Главный экран вашего телефона, вероятно, содержит множество приложений, основанных на контенте, таких как Netflix, Twitter, Facebook, Reddit и Evernote. Это очень популярные приложения, поддерживаемые гигантскими компаниями. Сейчас мы не говорим, что эти приложения работают на WordPress, но мы предполагаем, что *можно* создать приложение, похожее на это, с использованием WordPress в качестве фреймворка.

Во-вторых, как мы подробно рассмотрим в этой книге, WordPress — это отличная платформа для создания более интерактивных веб-приложений. Основной функцией, позволяющей выбрать WordPress в качестве основы, является API плагинов, который позволяет вам понять, как работает WordPress по умолчанию, и что-то изменить. Вам доступны не только тысячи плагинов в репозитории WordPress и других местах в Интернете, API плагинов позволяет вам написать собственные пользовательские плагины, чтобы WordPress делал что угодно с помощью PHP/MySQL.

WordPress не масштабируется

Некоторые из людей, которые так говорят, будут указывать на установку WordPress по умолчанию, работающую на хостинге нижнего уровня, и они отмечают, как сайт замедляется или "падает" при большой нагрузке, и, таким образом, приходят к выводу, что WordPress не масштабируется.

Или, может быть, когда мы предложили вам создать сайт, такой как Facebook, с использованием WordPress, вы справедливо насмехались над этой идеей.

Если вы намереваетесь создать приложение в масштабе Facebook, эта книга не для вас. Спросите своего технического директора, какая часть их бюджета в миллиард долларов выделена вашему приложению и каких инженеров вам нужно переманить из Google и Amazon, чтобы создать собственное решение.

В действительности многие сайты с большим трафиком работают на WordPress. WordPress.com работает на том же базовом программном обеспечении, что и любой сайт WordPress, и является одним из самых популярных сайтов в мире.

По мере расширения вашего приложения вам необходимо будет обновлять и заменять отдельные компоненты, чтобы соответствовать новому масштабу. Проблемы с масштабированием WordPress такие же, как и при масштабировании любого приложения: кэширование страниц и данных, более быстрая обработка вызовов базы данных и повышение производительности сети. Крупные сайты, такие как WordPress.com, TechCrunch и блог *New York Times*, стали использовать WordPress. Точно так же большинство уроков по масштабированию приложений PHP/MySQL в целом применимы и к WordPress. Мы подробно расскажем о масштабировании приложений WordPress в *главе 14*.

WordPress небезопасен

Как и для любого продукта с открытым исходным кодом, при использовании WordPress есть компромисс в отношении безопасности.

С одной стороны, поскольку WordPress очень популярен, он часто становится целью хакеров, ищущих уязвимости безопасности. А поскольку исходный код открытый, уязвимости будет легче обнаружить.

С другой стороны, поскольку WordPress является продуктом с открытым исходным кодом, вы узнаете, если эксплойты станут общедоступными, и кто-то другой, вероятно, выпустит исправление для вас.

Мы чувствуем себя более уверенно, зная, что есть множество людей, пытающихся использовать WordPress, и столько же людей, которые работают над тем, чтобы защитить WordPress от взломов. Мы не верим в "безопасность через неизвестность", если только в качестве дополнительной меры. Мы бы предпочли, чтобы дыры в безопасности нашего программного обеспечения появлялись открыто, а не оставались незамеченными до самого худшего момента.

В *главе 8* более детально рассматриваются вопросы безопасности, в том числе дан список рекомендаций по повышению безопасности установки WordPress и описаны способы безопасной разработки.

Плагины WordPress ужасны

API плагинов в WordPress и тысячи плагинов, которые были разработаны с его использованием, являются "секретным соусом" и, по нашему мнению, основной причиной того, что WordPress стал настолько популярным и настолько успешным в качестве веб-платформы.

Некоторые люди скажут: "Конечно, есть тысячи плагинов, но они все ужасны". Ну да, некоторые из плагинов *действительно* не очень полезны.

Но есть много плагинов, которые определенно не являются чепухой, например AppPresser, разработанный одним из авторов этой книги Брайаном Мессенленером. Если вам нравится WordPress для управления своим письменным контентом или

интернет-магазином, то плагин и платформа AppPresser — вот самый быстрый способ получить данный контент или сохранить в мобильном приложении.

Плагин Paid Memberships Pro, разработанный другим автором Джейсоном Коулманом, тоже хорош. Использование Paid Memberships Pro для управления биллингом и участниками позволит вам сосредоточить усилия по разработке на основной функциональности вашего приложения, а не на том, как интегрировать ваш сайт с платежной системой.

Многие плагины делают что-то очень простое (например, скрывают панель администратора от обычных пользователей), работают именно так, как рекламируется, и на самом деле вовсе не ужасны.

Темы и плагины, найденные в репозитории WordPress.org, тщательно проверяются сообществом на предмет безопасности и качества кода. Общеизвестно, что обзор на темы WordPress.org (**oreil.ly/WgTyD**) более строгий и всеобъемлющий, чем в других платформах. Проект Tide (**oreil.ly/iR94e***)* работает над добавлением автоматических тестов для плагинов и репозиториев тем, что приведет к увеличению качества плагинов и обновлений, а также позволит обнаружить проблемы совместимости и безопасности быстрее.

Даже дрянные плагины можно исправить, переписать или заимствовать для улучшения их работы. Иногда проще переписать плохой плагин, чем исправлять его. Однако в данном случае вы окажетесь все же на шаг ближе к цели, чем при написании с нуля.

Никто не заставляет вас использовать плагины WordPress, не проверяя их самостоятельно. Если вы создаете серьезное веб-приложение, то должны самостоятельно проверить код плагина, исправить его в соответствии со своими стандартами и продолжить разработку.

Когда не следует использовать WordPress

WordPress не является универсальным решением для любого приложения. В этом разделе описываются несколько случаев, в которых вы *не хотели бы* использовать WordPress для создания своего приложения.

Вы планируете лицензировать или продавать технологию своего сайта

WordPress использует Общедоступную лицензию GNU версии 2 (GPLv2), которая содержит ограничения на то, как вы распространяете программное обеспечение, созданное с его помощью. А именно, вы не можете ограничивать то, что люди делают с вашим программным обеспечением, когда вы продаете или распространяете его.

Это сложная тема, но основная идея заключается в том, что если вы продаете только свое приложение или предоставляете *доступ* к нему, то вам не нужно беспоко-

иться о GPLv2. Однако если вы продаете или распространяете исходный код вашего приложения, то GPLv2 будет применяться к распространяемому вами коду.

Например, если мы размещаем SchoolPress на наших собственных серверах и продаем учетные записи для доступа к приложению, это не считается распространением, а GPLv2 никак не влияет на наш бизнес.

Но если мы захотим разрешить школам устанавливать программное обеспечение для запуска на своих собственных серверах, то нам необходимо предоставить им исходный код. Это будет считаться актом распространения. Наши клиенты смогут на законных основаниях бесплатно раздавать наш исходный код, даже если мы изначально взяли с них плату за программное обеспечение. Мы должны учитывать лицензию GPLv2, которая не позволяет нам ограничивать то, что пользователи делают с кодом после его загрузки.

Имеется другая платформа, которая приведет вас к цели быстрее

Если у вас есть команда опытных разработчиков Ruby, то целесообразно создавать веб-приложение именно на Ruby. Если есть платформа, фреймворк или пакет, включающий 80% функций Ruby, необходимых для вашего веб-приложения, и WordPress не имеет ничего подобного, то вам, вероятно, следует предпочесть эту другую платформу.

Гибкость не важна для вас

Одна из главных особенностей сайта WordPress — возможность быстрой замены частей сайта в соответствии с вашими потребностями. Например, если "лайки" Facebook перестали приносить трафик, вы можете удалить плагин Facebook Connect и установить плагин для Pinterest.

Как правило, обновление вашей темы или замена плагинов на сайте WordPress будет быстрее, чем разработка функций с нуля на другой платформе. Однако в тех случаях, когда оптимизация и производительность важнее, чем возможность быстрого обновления приложения, лучшим выбором будет программирование нативного приложения или программирование прямо на PHP.

Если ваше приложение служит для выполнения *одной простой задачи*, то вы, скорее всего, захотите построить его на более низком уровне. Например, сервер лицензий Paid Memberships Pro представляет собой один файл JSON с дополнительной информацией и небольшим скриптом для проверки лицензионных ключей и доставки сжатых файлов. Джейсон создал этот сервер лицензий на PHP с большим объемом кэширования. Сервер лицензий функционирует на DigitalOcean Sroplet за 10 долларов в месяц и обслуживает более 80 тыс. сайтов, работающих под управлением Paid Memberships Pro.

Точно так же, если у вас есть ресурсы масштаба Facebook, то вы можете позволить себе создавать все вручную и использовать собственные компиляторы PHP-to-C и

нативные компоненты iOS, чтобы сократить время загрузки вашего веб-сайта и приложения на несколько миллисекунд.

Ваше приложение должно работать в режиме реального времени

Одним из потенциальных недостатков WordPress, о котором мы поговорим позже, является его зависимость от типичной архитектуры веб-сервера. В стандартной настройке WordPress пользователь посещает URL-адрес, который связывается с веб-сервером (например, Apache) по HTTP, запускает скрипт PHP для генерации страницы, а затем возвращает пользователю полную страницу.

Существуют способы улучшить производительность этой архитектуры с использованием методов кэширования и/или оптимизированных настроек сервера. Вы можете сделать WordPress асинхронным с помощью вызовов Ajax или доступа к базе данных с помощью альтернативных клиентов. Однако если ваше приложение должно работать в режиме реального времени и быть полностью асинхронным (например, приложение, похожее на чат, или многопользовательская игра), мы советуем хорошенько подумать, подойдет ли вам WordPress.

Многие разработчики WordPress, включая Мэтта Мулленвега, основателя и духовного лидера WordPress, понимают это ограничение. Все больше функциональности WordPress переносится в JavaScript, где вычисления могут быть перенесены в браузер, а фреймворки, такие как REACT, позволяют создавать высокоинтерактивные события. Новый редактор Gutenberg, добавленный в WordPress 5.0, является лучшим примером этого шага и свидетельствует о грядущих событиях. Но в данный момент вы столкнетесь с трудностями, пытаясь заставить WordPress работать асинхронно с той же производительностью, что и нативное приложение или что-то полностью построенное на Node.js либо других технологиях, специально предназначенных для приложений реального времени.

WordPress как фреймворк

Системы управления контентом, такие как WordPress, Drupal и Joomla, часто остаются вне обсуждения фреймворков, но на самом деле WordPress (в частности) действительно отлично подходит для того, для чего предназначены фреймворки: для быстрого создания приложений.

В течение нескольких минут вы можете настроить WordPress и получить полнофункциональное приложение с регистрацией пользователей, управлением сеансами, управлением контентом и панелью для мониторинга активности сайта.

Различные API, общие объекты и вспомогательные функции, описанные в этой книге, позволяют быстрее программировать сложные приложения, не беспокоясь об интеграции систем более низкого уровня.

На рис. 1.2 показан правый треугольник из презентации Мэтта Мулленвега "Состояние WordPress" 2013 года, изображающий устойчивую платформу WordPress со

слоем CMS, находящимся сверху, и приложение для ведения блогов, построенное поверх слоя CMS.

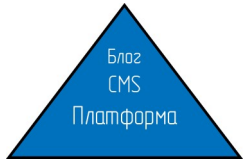


Рис. 1.2. Платформа WordPress

Реальность такова, что большая часть текущей кодовой базы WordPress поддерживает лежащую в основе платформу приложений. Думайте о каждом выпуске WordPress как о фреймворке приложения, который поставляется с образцом приложения для блога.

WordPress и фреймворки Framework-View-Controller

Модель-представление-контроллер (MVC — Model-View-Controller) — это общий шаблон проектирования, присутствующий во многих средах разработки программного обеспечения. Основные преимущества архитектуры MVC — повторное использование кода и разделение задач (SoC — Separation of Concerns). WordPress не задействует архитектуру MVC, но по-своему реализует повторное использование кода и SoC.

Здесь мы кратко расскажем об архитектуре MVC и о том, как она влияет на процесс разработки в WordPress. Если вы знакомы с основами MVC, этот раздел должен помочь вам понять, как подходить к разработке на WordPress аналогичным образом. На рис. 1.3 приведено типичное приложение на основе MVC. Конечный пользователь использует *контроллер*, манипулирующий состоянием приложения и данными через *модель*, которая затем обновляет *представление*, отображаемое пользователю. Например, в приложении блога пользователь может просматривать страницу последних сообщений (представление). Пользователь щелкает заголовок сообщения, осуществляя переход к новому URL (контроллеру), который загрузит данные поста (в модель) и отобразит один пост (другое представление).

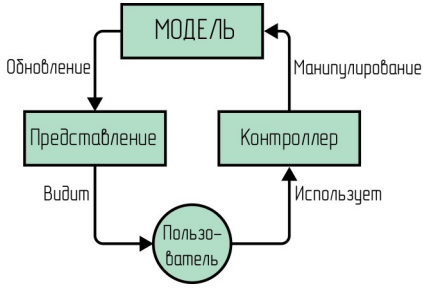


Рис. 1.3. Как работает MVC

Архитектура MVC поддерживает повторное использование кода, позволяя моделям, представлениям и контроллерам взаимодействовать. Например, и представление последних сообщений, и представление отдельных сообщений могут задействовать одну и ту же модель публикации при отображении данных публикации. Одни и те же модели могут служить в веб-интерфейсе для отображения сообщений и в бэкенде — для их редактирования. Архитектура MVC поддерживает SoC, позволяя дизайнерам сосредоточить свое внимание на представлениях, а программистам — на моделях.

Вы можете попробовать применить архитектуру MVC в WordPress. Есть целый ряд проектов, которые помогут вам сделать это. Однако мы считаем, что попытка привязать MVC к WordPress может привести к проблемам, если ядро WordPress не будет официально поддерживать MVC. Вместо этого мы предлагаем следовать "WordPress Way", как описано в этой книге.

Тем не менее, если вам интересно, WP MVC плагин (**bit.ly/wp-mvc**) находится в активной разработке и помогает вам на основе MVC Framework создавать плагины WordPress. Если вы не хотите или вам не нужен полный MVC, есть несколько способов привязать процесс MVC к WordPress.

Модели = плагины

В среде MVC код, который хранит базовые структуры данных и логику работы, находится в моделях. Именно здесь программисты будут проводить большую часть своего времени.

В WordPress плагины являются подходящим местом для хранения новых структур данных, сложной бизнес-логики и пользовательских определений типов записей.

Но это работает не всегда. Во-первых, многие плагины добавляют функциональность, подобную представлению, и содержат элементы дизайна (возьмите любой плагин, который добавляет виджет на ваши страницы). Во-вторых, формы и другие компоненты дизайна, используемые в инструментах WordPress, также обычно обрабатываются в плагинах.

Один из способов сделать SoC более понятным при добавлении в ваши плагины WordPress компонентов, отвечающих за дизайн, — создать папку templates или pages и поместить в нее свой код внешнего интерфейса. Обычная практика — позволить шаблонам (templates) переопределять шаблон, используемый плагином. Например, при работе WordPress с плагином Paid Memberships Pro вы можете скопировать папку с именем paid-memberships-pro/pages в папку с вашей активной темой, чтобы переопределить шаблоны страниц по умолчанию. (Этот метод переопределения шаблонов плагинов описан в *главе 4*.)

Представления = темы

В MVC-фреймворке код для отображения данных пользователю записывается в представлениях. Именно здесь дизайнеры и разработчики внешнего интерфейса будут проводить бóльшую часть своего времени.

В WordPress темы — это подходящее место для хранения шаблонного кода и логики.

Опять же, сравнение здесь не сопоставляется один к одному, но "views = themes" — хорошая отправная точка.

Контроллер = загрузчик шаблонов

В инфраструктуре MVC код для обработки пользовательского ввода (в виде URL-адресов или данных `$GET` или `$POST`) и определения того, какие модели и представления следует использовать для обработки запроса, хранится в контроллерах. Код контроллера обычно обрабатывается программистом и часто устанавливается один раз, а затем забывается. Программирование в приложении MVC происходит, по сути, в моделях и представлениях. Несмотря на это, контроллеры являются важной частью работы приложения.

В WordPress все запросы к страницам (если пользователи не обращаются к кэшированному файлу *.*html*) обрабатываются с помощью файла *index.php* в соответствии с иерархией шаблонов. Загрузчик шаблонов определяет, какой файл в шаблоне следует использовать для отображения страницы конечному пользователю. Например, *search.php* — для отображения результатов поиска, *single.php* — для отображения одного сообщения и т. д.

Поведение по умолчанию можно дополнительно настроить с помощью API-интерфейса WP_Rewrite (описанного в *главе 7*) и других хуков и фильтров. Информацию об иерархии шаблонов (**bit.ly/temp-hier**) вы можете найти в справочнике по WordPress. Мы более подробно рассмотрим иерархию шаблонов в *главе 4*.

Для лучшего понимания того, как работают фреймворки MVC, фреймворк PHP Yii (**bit.ly/yii-guide**) содержит большой ресурс, объясняющий подробно, как лучше всего использовать его архитектуру MVC[3].

Подробнее о том, как разрабатывать веб-приложения на основе WordPress, читайте далее в этой книге.

Анатомия приложения WordPress

В данном разделе опишем приложение, которое мы создали как дополнение к этой книге: SchoolPress. Мы расскажем о предполагаемой функциональности SchoolPress, о том, как оно работает, и кто будет им пользоваться, и, что наиболее важно для этой книги, о том, как каждая часть приложения построена в WordPress.

Не беспокойтесь, если вы не понимаете некоторые из следующих терминов. В последующих главах мы рассмотрим всю введенную здесь терминологию более подробно. Когда это возможно, мы указываем на главу, которая соответствует обсуждаемой функции.

[3] Yii – PHP-фреймворк на основе MVC. Другие платформы PHP, такие как Laravel (**laravel.com**), более популярны среди разработчиков WordPress и сообщества PHP в целом, но документация, связанная с MVC, на сайте Yii особенно хорошо написана.

Работа с API-интерфейсами WordPress, объектами и вспомогательными функциями

В этой главе мы обратим внимание на несколько API, объектов и вспомогательных функций WordPress, которые не рассматриваются в других главах книги, но все еще являются важными частями арсенала разработчика WordPress.

API шорткодов

Шорткоды — это специально отформатированные фрагменты текста, с помощью которых можно вставлять динамические элементы в ваши публикации, страницы, виджеты и другие области статического содержимого.

Шорткоды бывают трех основных видов:

1. Единичный шорткод вроде `[myshortcode]`
2. Шорткоды с атрибутами, как `[myshortcode id = "1" type = "text"]`
3. Заключающие шорткоды, например так `[myshortcode id = "1"] ... некоторый контент здесь ... [/ myshortcode]`

В *главе 3* мы поделились примером того, как использовать шорткоды для добавления произвольного контента в пост или страницу WordPress. В примере, рассмотренном далее, как и раньше, мы просто заменили шорткод на наше содержимое. Вы также можете добавить атрибуты к шорткоду, чтобы повлиять на функцию обратного вызова, обрабатывающую его, или обернуть некоторый контент в пару открывающих и закрывающих шорткодов, чтобы отфильтровать некоторый конкретный контент.

Основной этап создания шорткодов — определение функции обратного вызова для вашего шорткода с помощью функции `add_shortcode()`. Любые атрибуты добавляются в массив, который передается обратному вызову в качестве первого параметра `$atts`. Произвольное вложенное содержимое передается обратному вызову в качестве второго параметра — `$content`.

Следующий пример создает шорткод с именем `msg` и задействует атрибуты и вложенный контент.

Пример

```
<?php
/*
  Обратный вызов для шорткода [msg]
  Пример: [msg type = "error"] Это сообщение об ошибке. [/msg]
  Вывод:
  <div class="message message-error">
      <p> Это сообщение об ошибке. </p>
  </div>
*/
function sp_msg_shortcode($atts, $content)
{
  // атрибуты по умолчанию
  extract(shortcode_atts(array(
      'type' => 'information',
    ), $atts));
  $content = do_shortcode($content); // разрешаем вложенные шорткоды
  $r = '<div class="message message-' .
    $type . '"><p>' . $content . '</p></div>';
  return $r;
}
add_shortcode('msg', 'sp_msg_shortcode');
?>
```

Обратите внимание, что содержимое, которое вы хотите отобразить, возвращается из функции обратного вызова, а не отражается в буфере вывода. Это связано с тем, что фильтр шорткода обычно запускается до того, как какой-либо контент будет помещен на экран. При наличии каких-либо эхо-вызовов внутри этой функции, вывод был бы показан вверху страницы, а не там, где вы хотите.

Атрибуты шорткода

В предыдущем примере продемонстрирован еще один важный момент — задание атрибутов по умолчанию. Функция `shortcode_atts()` принимает три параметра: `$pairs`, `$atts` и `$shortcode`.

- `$pairs` — массив атрибутов по умолчанию, где каждый ключ — это имя атрибута, а каждое значение — значение атрибута.
- `$atts` — это аналогичный массив атрибутов, обычно получаемый прямо из параметра `$atts`, переданного в функцию обратного вызова шорткода. Функция `shortcode_atts()` объединяет атрибуты по умолчанию и переданные атрибуты в один массив.
- `$shortcode` — необязательный параметр. Если установлено совпадение с именем шорткода, запускается фильтр `shortcode_atts_{shortcode}`, который может использоваться другими плагинами (или чем-то еще) для переопределения атрибутов по умолчанию.

Результаты функции shortcode_atts() затем передаются в PHP-функцию extract(), которая создает переменную в локальной области видимости для каждого ключа в массиве атрибутов.

Таким образом, переменная $type в нашем примере доступна для остальной части функции и содержит либо значение по умолчанию для сообщения, либо любое другое значение, установленное в самом шорткоде.

Вложенные шорткоды

Наконец, мы пропускаем внутреннее содержимое $content через функцию do_shortcode(), чтобы включить вложенные шорткоды. Если у вас был шорткод [help_link], который генерировал ссылку на вашу документацию, в зависимости от того, на каком разделе сайта вы работали, или под каким типом пользователя вы вошли в систему, вы можете использовать этот шорткод в шорткоде [msg]:

```
[msg type="error"]
        Произошла ошибка. Используйте следующую ссылку для справки: [help_link].
[/msg]
```

Пока функция обратного вызова для шорткода [msg] передает свои результаты через функцию do_shortcode(), внутренний шорткод [help_link] будет отфильтрован как предполагалось.

Несмотря на то, что вложенные шорткоды разных типов будут работать, вложение одного и того же шорткода внутрь самого себя вызовет ошибку. Парсер регулярных выражений (regex), который извлекает шорткоды из контента, спроектирован для увеличения скорости выполнения. Парсер должен проходить по содержимому только один раз. Для обработки вложенных шорткодов одного и того же типа потребовалось бы несколько проходов через контент, что замедлило бы алгоритм. Возможные решения этой проблемы: 1) избежать вложения одного и того же шорткода в самого себя; 2) задать шорткодам, которые ссылаются на одну и ту же функцию обратного вызова, разные имена; 3) написать собственный синтаксический анализатор регулярных выражений для вашего шорткода и проанализировать шорткоды самостоятельно.

Функция do_shortcode() позволяет также применять шорткоды к настраиваемым полям, содержимому, извлеченному из пользовательских таблиц, или другому содержимому, которое еще не проходило через фильтр the_content. В большинстве случаев вне функций самого обратного вызова шорткода, более целесообразно использовать функцию apply_filters("the_content", $content), которая будет применять все фильтры при срабатывании хука the_content, включая фильтр шорткодов, как показано в следующем примере.

Пример

```
<?php
global $post;
$sidebar_content = $post->sidebar_content;
?>
```

```
<div class="post">
  <?php the_content(); ?>
</div>
<div class="sidebar">
  <?php
  //echo do_shortcode($sidebar_content);
  echo apply_filters('the_content', $sidebar_content);
  ?>
</div>
```

Удаление шорткодов

Как и в случае с событиями и фильтрами, вы можете удалить зарегистрированные шорткоды, чтобы они не применялись к определенному сообщению или к содержимому, которое вы передаете непосредственно в код `do_short()` или через фильтр `the_content`. Функция `remove_shortcode()` принимает имя шорткода в качестве единственного параметра и отменяет регистрацию указанного шорткода. Функция `remove_all_shortcodes()` отменяет регистрацию всех шорткодов.

Убедитесь, что вызов функции `remove_shortcode()` выполняется достаточно поздно при работе WordPress, чтобы шорткод, который вы хотите удалить, уже был добавлен. Например, если плагин добавляет шорткод во время действия хука `init` с приоритетом 10, то вам нужно поместить вызов `remove_shortcode()` во время действия `init` с приоритетом 11 или выше или через другой хук, который срабатывает после `init`.

Массив зарегистрированных шорткодов хранится в глобальной переменной `$shortcode_tags`. Иногда полезно сделать копии этой переменной или отредактировать ее напрямую. Например, если вы хотите исключить конкретные шорткоды из определенного фрагмента содержимого, то можно сделать резервную копию всех шорткодов, удалить ненужные шорткоды, применить их, а затем восстановить исходный список шорткодов, что демонстрирует следующий пример.

Пример

```
// Делаем копию оригинального списка шорткодов
global $shortcode_tags;
$original_shortcode_tags = $shortcode_tags;

// удаляем шорткод [msg]
unset($shortcode_tags['msg']);

// выполнеям шорткоды и вывод
$content = do_shortcode($content);
echo $content;

// восстанавливаем оригинальные шорткоды
$shortcode_tags = $original_shortcode_tags;
```

Другие полезные функции, связанные с шорткодами

Укажем еще несколько функций, полезных при работе с шорткодами:

- `shortcode_exists($tag)` — проверяет, был ли зарегистрирован шорткод `$tag`;
- `has_shortcode($content, $tag)` — проверяет, присутствует ли шорткод `$tag` в переменной `$content`;
- `shortcode_parse_atts($text)` — извлекает атрибуты из шорткода. Это делается за вас при парсинге любого шорткода, но эта функция может вызываться напрямую, если вы хотите извлечь атрибуты из другого текста, такого как теги HTML или другие шаблоны;
- `strip_shortcodes($text)` — удаляет все шорткоды из переменной `$text` и заменяет их пустым текстом вместо выполнения функции обратного вызова.

Более подробную информацию об API шорткодов можно найти в Кодексе WordPress (**bwawwp.com/shortc-api**).

API виджетов

Виджеты позволяют отображать фрагменты кода и содержимое в различных областях на вашем сайте WordPress. Наиболее типичные случаи — это добавление виджетов на боковую панель или область нижнего колонтитула. Вы всегда можете жестко закодировать эти разделы на веб-сайте, но наличие виджетов позволяет вашим разработчикам перетаскивать их из одной области в другую или изменять свои настройки через страницу виджетов на панели администратора. WordPress поставляется со множеством встроенных виджетов, включая базовый текстовый виджет, показанный на рис. 7.1.

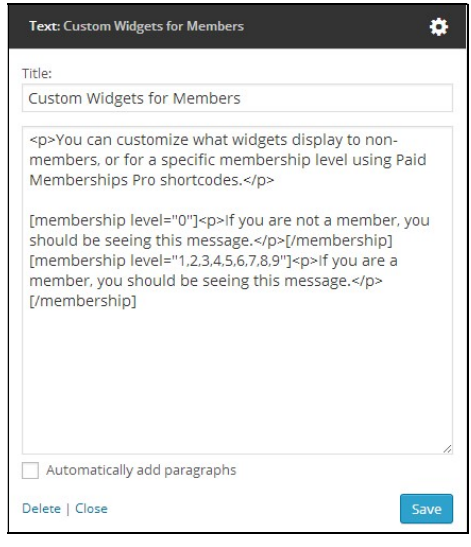


Рис. 7.1. Настройки текстового виджета

Множество плагинов также включает в себя виджеты для отображения различного контента. Здесь мы не будем использовать и стилизовать виджеты, так как это хорошо описано на странице Кодекса WordPress, посвященной виджетам (**bwawwp.com/widgets-codex**), но рассмотрим, как добавлять виджеты и области виджетов в ваши плагины и темы.

Пользовательский интерфейс страницы виджетов в админ-панели претерпел изменения в WordPress версии 3.8; однако функции и вызовы API для добавления новых виджетов программными средствами не должны сильно изменяться.

Прежде чем добавить свой собственный виджет

Прежде чем приступить к разработке нового виджета, стоит потратить некоторое время, чтобы посмотреть, не подойдет ли вам существующий виджет. Если вы проявите творческий подход, то сможете избежать создания нового виджета.

Найдите в репозитории плагины, в которых уже есть нужный вам виджет. Если это так, дважды проверьте код и посмотрите, будет ли он работать.

Текстовые виджеты могут использоваться для добавления произвольного текста в пространство виджетов. Таким образом вы также можете встроить код JavaScript или добавить шорткод в текстовую область и с его помощью реализовать нужную вам функциональность (возможно, вы создали шорткод уже для другой цели) вместо создания нового виджета.

Если ваш виджет отображает список ссылок, возможно, имеет смысл создать меню из этих ссылок и использовать виджет "Custom Menu", встроенный в WordPress. Другие виджеты, отображающие последние публикации из категории, часто работают с CPT и таксономиями либо сразу "из коробки", либо требуют совсем немного изменений.

Если вам нужен совершенно новый виджет, в следующем разделе описываются необходимые шаги.

Добавление виджетов

Чтобы добавить еще один виджет в WordPress, необходимо создать новый класс PHP для виджета, который расширяет класс `WP_Widget` WordPress. Рекомендуем вам ознакомиться с кодом класса `WP_Widget`, находящимся в файле в wp-includes/widgets.php. Комментарии в коде объясняют, как класс работает и какие методы вы должны переопределить, чтобы создать свой собственный класс виджетов. Существуют четыре основных метода, которые вы должны переопределить для класса виджета, что ясно показано в следующем примере кода со страницы Кодекса WordPress, посвященной API виджетов (**oreil.ly/ojY06**).

Пример

```
/*
  Взято со страницы Кодекса API виджетов по адресу:
  http://codex.wordpress.org/Widgets_API
```

Проект Gutenberg, блоки и кастомные типы блоков

Когда в январе 2017 года был предложен новый редактор для WordPress, Мэтт Мулленвег написал следующее (**oreil.ly/ClAYx**):

"Данный редактор призван обеспечить новый опыт создания новых страниц и постов, который упрощает написание объемных постов и предлагает "блоки" для легкого создания тех вещей, которые сегодня требуют использования коротких кодов, кастомного HTML-кода или малопонятных встроенных представлений".

Не прошло и двух лет, как редактор блоков Gutenberg был включен в состав версии 5.0 системы WordPress, принеся с собой новый способ редактирования постов и разработки с помощью WordPress.

"Gutenberg" — это кодовое название исходного проекта по разработке редактора блоков. Сегодня мы предпочитаем говорить "*редактор блоков*" или просто "*редактор системы WordPress*", однако часто используется и название "Gutenberg", под которым подразумевается сам редактор.

Чтобы помочь и пользователям, и разработчикам в освоении возможностей редактора Gutenberg, команда его создателей составила "Руководство по редактору блоков" (**oreil.ly/ R8JyX**). Это компактная, хорошо написанная и постоянно дорабатываемая документация, которую обязательно следует прочитать. Мы рекомендуем вам ознакомиться с ней прямо сейчас, а затем вернуться к этому месту. В представленном далее материале будут даны ссылки на некоторые разделы этого руководства.

В данной главе мы кратко рассмотрим основные особенности редактора блоков, создадим в качестве отправной точки простейший блок, после чего чуть подробнее коснемся тех возможностей, которые представляют наибольший интерес при разработке приложений.

Редактор блоков реализован главным образом с помощью JavaScript-кода (использующего пользовательский интерфейс) и администрируется как проект Node.js. Хотя руководство по редактору Gutenberg может оказать вам реальную помощь в программировании блоков, в представленных в нем примерах очень сложно понять, к чему относится та или иная часть кода — к JavaScript-коду, фреймворку React или редактору Gutenberg. Понимание того, как взаимодействуют друг с другом эти технологии, может оказаться очень полезным при разработке расширений для WordPress. В своих примерах мы постарались как можно яснее показать, откуда берется тот или иной код.

Редактор системы WordPress

Базовым элементом нового редактора системы WordPress являются блоки, которые располагаются один за другим и представляют абзацы, заголовки, списки, изображения и более сложные компоненты. Некоторые типы блоков, например блоки группы и колонки, могут содержать вложенные блоки. Вы можете менять расположение блоков, перетаскивая их в окне редактора (рис. 11.1).

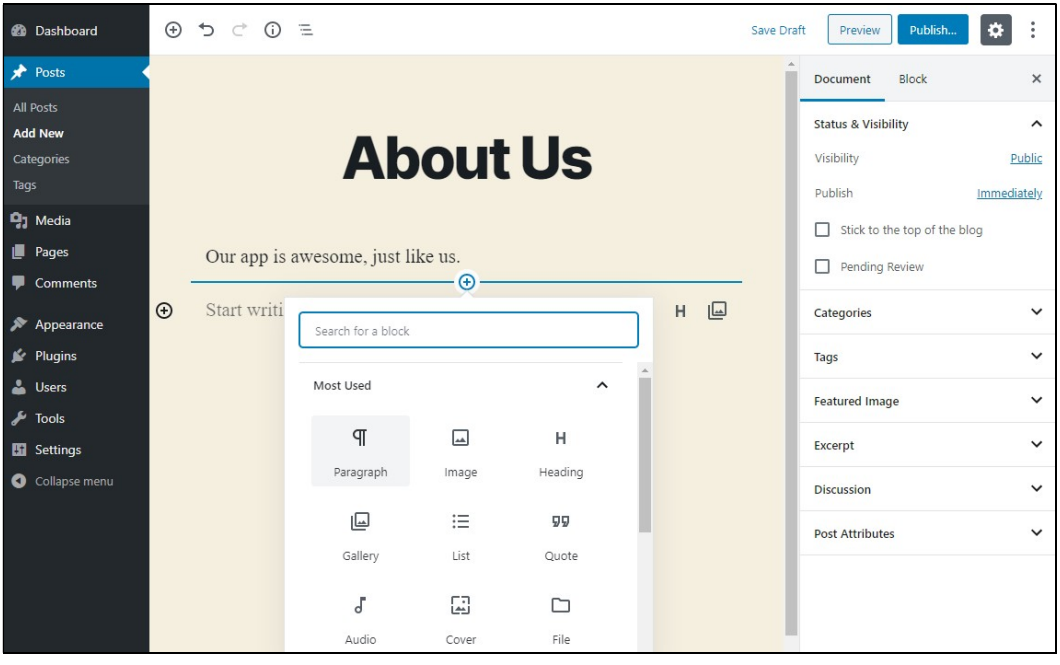


Рис. 11.1. Редактор блоков системы WordPress

Вы можете добавлять блоки, выполняя щелчок по кнопке + в пользовательском интерфейсе или вводя символ / и имя блока в пустом месте в окне редактора. Нажатие клавиши ввода внутри блока абзаца ведет к созданию нового блока абзаца. Некоторые типы блоков создаются автоматически при вводе определенных символов разметки внутри пустого блока. Так, символ * начинает маркированный список, а символы ## — вторичный заголовок[1].

Каждый блок редактируется по отдельности. Для этого нужно щелкнуть по блоку, чтобы передать ему фокус и переключиться в окно редактора. При этом вкладка **Block** (Блок) правой боковой панели будет содержать настройки, специфичные для редактируемого блока. Выполнив щелчок по другому блоку или по пустому месту в окне редактора, вы снова вернетесь к клиентской части приложения.

[1] Полный список форматирующих кодов и полезных клавишных комбинаций можно найти на сайте WordPress (**oreil.ly/0StKa**).

Редактор системы WordPress будет в максимально возможной степени зеркалировать клиентскую часть. Хотя для упрощения редактирования блоки могут отображаться с метками-заполнителями и некоторыми отличиями в форматировании, разработчикам рекомендуется делать так, чтобы внешний вид создаваемых ими блоков в окне редактора имел как можно больше сходства с их видом в клиентской части.

Плагин Classic Editor

Если на вашем сайте есть плагины или функции, которые требуют наличия классического редактора, или вам просто больше нравится старый редактор в стиле TinyMCE, вы можете установить плагин Classic Editor, предлагаемый по адресу **oreil.ly/BqQ9K**. Этот плагин позволяет выборочным или глобальным образом использовать классический редактор для редактирования любого поста или страницы. Как указывается в файле *README*, "Classic Editor является официальным плагином WordPress, и будет обеспечен полной поддержкой и обслуживанием, как минимум, до 2022 года или до тех пор, пока это будет необходимо".

Мы рекомендуем плагин Classic Editor лишь в крайнем случае. Несмотря на то, что его планируется поддерживать вплоть до 2022 года, редактор блоков постепенно станет неотъемлемой частью панели администратора WordPress, и новые плагины и темы будут использовать его возможности.

Блоки для контента и дизайна

Большинство предлагаемых типов блоков имеют отношение к контенту и дизайну. Они позволяют создавать насыщенные макеты дизайна, не прибегая к написанию кода. Поскольку обсуждение популярных сегодня блоков для представления разделителей, каруселей или галерей выходит за рамки данной книги, мы лишь отметим, что у вас будет достаточный выбор таких блоков.

Блоки для представления функциональности

Часть блоков служит главным образом для представления функциональности сайта или приложения на базе WordPress. Так в плагине Paid Memberships Pro используется блок членства, который накладывает ограничения на все вложенные блоки в зависимости от уровня членства. Со временем плагины со сложными пользовательским интерфейсом клиентской части будут комплектоваться блоками, обеспечивающими большую степень контроля над макетом и настройками пользовательского интерфейса. Так, вы могли бы выполнять тонкую настройку макета каталога в BuddyPress с помощью блоков и их параметров.

Создание собственных блоков

Как бы ни было приятно использовать готовые блоки, создавать собственные блоки еще приятнее... или, скажем, почти всегда приятнее. Иногда это довольно трудно, однако давайте начнем с простейшего примера блока, после чего рассмотрим что-то посложнее.

Пример простейшего блока

Чтобы добавить блок в редакторе, нужно, как минимум, вызвать функцию `wp.blocks.registerBlockType()` в JavaScript-коде, запускаемом на странице редактора, что иллюстрирует следующий пример.

Пример

```
wp.blocks.registerBlockType('bwawwp/minimal', {
  title: 'Minimal Block Example',
  category: 'common',
  edit() {
    return 'Minimal block editor content.';
  },
  save() {
    return 'Minimal block frontend content.';
  },
});
```

Полное описание и список параметров этой функции можно найти в разделе "Регистрация блоков" Руководства по редактору блоков (**oreil.ly/Ydx9m**).

Наш простейший блок передает в функцию имя блока и массив аргументов. Блокам следует давать уникальные имена, указывая в качестве префикса пространство имен или слаг того плагина, который загружает блок. Имена должны начинаться с буквы и могут содержать *строчные* буквы и цифры, а также дефисы.

Наш простейший блок также передает в функцию заголовок (`title`) и имя категории (`category`). В качестве заголовка допускается любая описательная строка, которая упростит поиск блока в общем списке. Чуть позже будет показано, как можно добавить новую категорию для группировки родственных блоков. В ядро WordPress включены следующие категории: *common* (общие), *formatting* (форматирование), *layout* (макет), *widgets* (виджеты) и *embed* (встроенные).

"Рабочими лошадками" функции `registerBlockType` являются атрибуты `edit` и `save`, в качестве которых передаются функции для вычисления и возвращения структуры блока. Функция `edit` используется при отображении блока в редакторе (рис. 11.2). Функция `save` вызывается при сохранении поста, перед сериализацией блоков в переменную в `post_content`.

В данном простейшем примере блока мы возвращаем лишь простую строку. Обратите внимание, что при наличии в этой строке HTML-кода он будет санирован и преобразован в HTML-сущности. Для генерирования разметки, содержащей ваши блоки, вам потребуется воспользоваться функцией `wp.element.createElement()`.

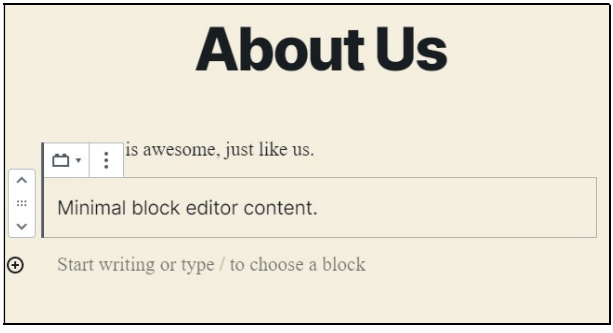


Рис. 11.2. Простейший блок в редакторе

Мы можем создать простой плагин с одним каталогом и двумя файлами, выполняющими загрузку нашего блока: block.js и minimal-block-example.php.

В следующем примере показано, как выглядит код файла minimal-block-example.php, выполняющий подключение файла block.js.

Пример

```
/**
* Имя плагина: Простейший пример блока
*/
function enqueue_min_block() {
        wp_enqueue_script(
                'minimal-block',
                plugins_url('block.js', __FILE__),
                array('wp-blocks')
       );
}
add_action('enqueue_block_editor_assets', 'enqueue_min_block');
```

Данный код аналогичен коду для подключения другого JavaScript-кода, приводившемуся в *главе 9*. В рассмотренном случае вместо обычного хука действия `wp_enqueue_scripts` мы подключаемся к хуку действия `enqueue_block_editor_assets`. Также обратите внимание, что в качестве зависимости для JavaScript-кода нашего блока указывается пакет `wp-blocks`. По мере расширения функциональности блоков нам потребуется подключать здесь и другие пакеты, имеющие отношение к редактору Gutenberg.

Приведенный пример также демонстрирует минимальные требования к плагину. Единственный обязательный параметр в заголовке — имя плагина. Остальные параметры являются опциональными, хотя, как было показано в *главе 3*, присутствие некоторых из них будет крайне желательным.

Теперь мы рекомендуем вам снова обратиться к документации, представленной на сайте WordPress.org, и ознакомиться с разделом "Создание вашего первого типа блоков" (**oreil.ly/7qLtP**). В этом разделе освещается добавление стилей, использование редактируемых полей, добавление панелей инструментов и настроек, а также создание динамически обновляемых блоков.

Использование кастомных блоков для разработки интерфейсов приложений

Допустим, что преподавателям, работающим с нашим приложением SchoolPress, иногда требуется создавать новые домашние задания. Конечно, они могут использовать Microsoft Word или Adobe Acrobat или просто ввести нужный текст в виде обычного поста WordPress. Однако нам нужно нечто хорошо согласующееся с остальной частью создаваемого нами приложения, позволяющее сохранять ответы на задания в базе данных для обеспечения отчетности и другой функциональности.

Мы можем создать форму, написав кастомный PHP-код. Можно создать что-то более динамичное с помощью фреймворка React, обеспечив сохранение данных в WordPress посредством WordPress REST API. Также можно использовать плагин Advanced Custom Fields в сочетании с кастомным типом поста для домашнего задания, обеспечивающим надлежащую компоновку необходимых данных. Как и в случае многих других задач разработки WordPress-приложений и программной разработки в целом, в данном случае имеется несколько рационально обоснованных вариантов решения.

Наибольший интерес при этом вызывает вариант, подразумевающий создание кастомного типа поста для домашнего задания, кастомных типов и шаблонов блоков. Взяв за основу редактор блоков, мы будем базироваться на интерфейсе, уже знакомом для наших пользователей. Мы можем определить свои кастомные типы и шаблоны блоков таким образом, чтобы данные домашнего задания всегда представлялись в нужном для нашего WordPress-приложения структурированном формате.

Активация редактора блоков для кастомных типов постов

В версии 5.3 системы WordPress для кастомных типов постов по умолчанию задан классический редактор. Со временем для всех кастомных типов постов по умолчанию будет назначен новый редактор блоков. На данный момент для использования редактора блоков для кастомных типов постов необходимо его явно активировать при регистрации кастомных типов постов.

Давайте снова вернемся к нашему примеру с кастомным типом поста для домашнего задания. В следующем примере показана настройка редактора блоков.

Электронная коммерция

В какой-то момент у вас может возникнуть желание взимать плату за доступ к вашему приложению или принимать какие-нибудь другие платежи на своем сайте. В этой главе мы пройдемся по лучшим плагинам для электронной коммерции и платных подписок и дадим вам несколько подсказок о том, на чем следует основывать свой выбор. Мы также рассмотрим пошаговую инструкцию по настройке типичной системы платного доступа в модели SaaS (Software-as-a-Service — программное обеспечение как услуга).

Выбор плагина

У WordPress есть один плагин, который является чуть ли не синонимом электронной коммерции: WooCommerce. Начиная с 2014 года, когда было опубликовано первое издание этой книги, WooCommerce стал доминирующей платформой в данной области, и не только для WordPress. Здесь мы представим краткое описание этого плагина и рассмотрим несколько хуков и фильтров, которые могут заинтересовать разработчиков приложений.

Несмотря на то, что WooCommerce — отлично написанный и поддерживаемый продукт, в некоторых случаях более уместными могут оказаться и другие плагины. В связи с этим будут рассмотрены Paid Memberships Pro (плагин с акцентом на платные подписки) и Easy Digital Downloads (плагин с акцентом на виртуальные товары).

Все плагины, представленные в этой главе, обладают следующими возможностями:

- интеграция с несколькими платежными системами;
- безопасные формы оплаты;
- сохранение информации о заказе;
- перечисление продуктов (уровней подписки) с ценами.

Уникальные особенности каждого вида плагинов для электронной коммерции будут освещены в следующих разделах.

WooCommerce

Сложно подсчитать, какую долю рынка занимает платформа электронной коммерции[1], но, по всей видимости, как минимум 20% *всех* сайтов, которые занимаются

[1] Что мы понимаем под "платформой электронной коммерции"? На чем должна быть основана рыночная доля: на количестве сайтов или продаж? Каким образом следует учитывать готовые удаленные решение, такие как Shopify или Amazon?

торговлей, основаны на WooCommerce[2]. Если верить WordPress.org (**oreil.ly/odzOF**), этот плагин был загружен свыше 47 миллионов раз на более чем 4 миллионах активных сайтов. Как бы вы ни считали, WooCommerce используется на огромном количестве ресурсов. Вот почему компания Automattic купила WooCommerce еще в 2015 году, и именно поэтому данный продукт будет в центре нашего внимания.

WooCommerce, как и другие "корзины покупок", обладает следующими возможностями:

- пользовательские типы постов для продуктов;
- навигация по продуктам;
- возможность искать по продуктам;
- возможность купить сразу несколько продуктов;
- вычисление стоимости доставки с возможностью указания адреса;
- поддержка пользовательских правил для налогов.

Документацию о том, как настроить WooCommerce, и многое другое можно найти на сайте WooCommerce (**docs.woocommerce.com/**).

Плагин и расширения WooCommerce

Плагин WooCommerce доступен бесплатно в репозитории WordPress.org. Его основная часть содержит все, что необходимо для определения товаров и цен, а также для обработки платежей через PayPal. Недавно в бесплатном доступе появилось расширение Stripe Payment Gateway для WooCommerce, которое является самым популярным средством работы с кредитными картами непосредственно на вашем сайте, но аналогичные расширения существуют практически для любого способа оплаты.

У WooCommerce есть десятки платных и бесплатных расширений, которые улучшают основной плагин, интегрируют его со сторонними рекламными сервисами, добавляют новые типы товаров, предоставляют поддержку дополнительных служб доставки или упрощают управление магазином. Два самых популярных из них — WooCommerce Subscriptions (**oreil.ly/PqCwh**; позволяет запоминать повторяющиеся платежи) и WooCommerce Memberships (**oreil.ly/ 8lf8C**; тоже запоминает повторяющиеся платежи, но также умеет ограничивать доступ к контенту в зависимости от уровня подписки). Специально для этого сценария разработан плагин Paid Memberships Pro, о котором мы поговорим чуть позже, но если вы уже работаете с WooCommerce или вам нужная какая-то другая функция, с которой он хорошо справляется, перечисленные расширения будут хорошим выбором.

Изменение поведения WooCommerce с помощью хуков

WooCommerce (как и любой хороший плагин для WordPress) имеет огромное количество хуков с действиями и фильтрами, с помощью которых можно влиять на то,

[2] Согласно оценкам сервиса BuiltWith (**oreil.ly/sprAP**), на момент написания этой книги из миллиона самых популярных сайтов, занимающихся электронной коммерцией, 21% используют WooCommerce.

как он работает на вашем сайте. Их полный список можно найти по адресу **oreil.ly/Kk0ie**.

Далее приводятся примеры того, что можно сделать с помощью некоторых типичных хуков WooCommerce.

- *Подготовка распродажи в рамках всего сайта*. Глобальную распродажу в магазине WooCommerce можно организовать и с использованием готовых плагинов, но, поскольку вы не лыком шитый разработчик, вам, возможно, захочется сделать это самостоятельно с помощью самописного кода. В следующем примере устанавливается скидка в размере 10% для любого товара, который не участвует ни в какой другой распродаже.

Пример

```
// Устанавливаем скидку в 10% от обычной цены,
// если товар еще не находится в распродаже
function my_get_sale_price($sale_price, $product) {
        if(empty($sale_price)) {
                $sale_price = $product->get_regular_price() * .9;
    $product->set_price($sale_price);
        }

        return $sale_price;
}
add_filter('woocommerce_product_get_sale_price', 'my_get_sale_price',
  10, 2);
add_filter('woocommerce_product_variation_get_sale_price',
  'my_get_sale_price', 10, 2);
```

В WooCommerce у товаров есть как "обычная", так и "распродажная" цена: `regular_price` и `sale_price`. Итоговая вычисленная цена называется просто price. Для установки распродажной цены предусмотрен хук `woocommerce_product_get_sale_price`, но для поддержки нескольких разновидностей товаров (например, футболок размеров S, M и L) вам также понадобится хук `woocommerce_product_get_variation_sale_price`.

Обратите внимание на строчки с вызовом `$product->set_price($sale_price)`. Вычисляемая цена не обновляется автоматически после возвращения распродажной цены из этой функции обратного вызова; мы должны делать это вручную. Параметр `$product` передается в функцию по ссылке, поэтому обновление, которое здесь выполняется, изменяет товар и за пределами фильтра.

- *Автоматическое завершение заказов*. По умолчанию новому заказу, который создается на странице оплаты, присваивается статус "в ожидании", и администратор сайта должен его обработать. Для физических товаров *обработка* обычно подразумевает упаковку и отправку заказанного товара с последующим присвоением заказу статуса "завершен".

Заказы виртуальных товаров имеет смысл автоматически делать "завершенными". Это экономит вам лишний щелчок мыши и позволяет плагину WooCommerce быстрее среагировать на завершение заказа — отправить электронные письма, обратиться к соответствующим API-интерфейсам и т.д. Код следующего примера перебирает товары в корзине после оформления заказа и, если все они виртуальные, автоматически делает заказ завершенным.

Пример

```
function autocomplete_virtual_orders($order_id) {
        // Получаем существующий заказ
        $order = new WC_Order($order_id);

        // подразумевается, что включено автозавершение
        $autocomplete = true;

        // проходимся по заказанным товарам
        if (count($order->get_items()) > 0) {
          foreach ($order->get_items() as $item) {
            if($item['type'] == 'line_item') {
              $_product = $order->get_product_from_item($item);
              if(!$_product->is_virtual()) {
                // в корзине найден невиртуальный товар
                $autocomplete = false;
                break;
                                }
                        }
              }
        }

        // при необходимости меняем статус
        if(!empty($autocomplete)) {
                $order->update_status('completed', 'Autocompleted.');
        }
}
add_filter('woocommerce_thankyou', 'autocomplete_virtual_orders');
```

Paid Memberships Pro

Paid Memberships Pro и другие подобные плагины предназначены для приема платежей за доступ к приложению или сайту WordPress по подписке. Эти плагины включают в себя такие возможности:

- регулярная плата за подписку;
- средства блокирования доступа к контенту в зависимости от уровня подписки.

Документацию о том, как настроить Paid Memberships Pro, и многое другое можно найти на веб-сайте по адресу **oreil.ly/mrlLX**.

За что нам нравится Paid Memberships Pro

Плагин Paid Memberships Pro был разработан соавтором этой книги, Джейсоном Коулманом, но это далеко не единственное его преимущество. Это единственный плагин WordPress для подписок, который полностью распространяется под лицензией GPL (General Public License) и доступен бесплатно в репозитории WordPress. Другие плагины предоставляют часть своих возможностей либо в виде платных модулей, либо в отдельных платных версиях.

Весь код Paid Memberships Pro находится в публичных репозиториях на GitHub, и в его разработке может принимать участие кто угодно. Как и WooCommerce, он поддерживает хуки и фильтры для изменения стандартного поведения.

Почти у любого сайта с платными подписками есть свой собственный способ подсчета стоимости перехода на более высокий уровень подписки, специальных предложений или того, как и когда следует ограничивать доступ к контенту. У Paid Memberships Pro нет длиннющей страницы настроек; вместо этого данный плагин предлагает тщательно продуманные хуки и фильтры, которые позволяют создать практически любую модель ценообразования или ограничения контента с помощью всего нескольких строчек кода.

Еще одно ключевое отличие Paid Memberships Pro от аналогичных плагинов состоит в наличии собственной таблицы с уровнями подписок и информацией о том, как они связаны с пользователями и заказами. Некоторые плагины используют для определения уровней подписки встроенные в WordPress пользовательские роли. В некоторых сайтах с платными подписками (см. *главу 6*) пользовательские роли имеют большое значение, но в целом их лучше не привязывать к уровням подписки; таким образом одни зарегистрированные пользователи смогут быть администраторами, а другие подписчиками. Если же вам нужно назначать пользовательские роли в зависимости от уровня подписки, это можно легко сделать и в Paid Memberships Pro. Пример этого будет показан чуть позже в этой главе.

Позже вы сможете познакомиться с несколькими примерами использования Paid Memberships Pro. Но прежде чем переходить к общим концепциям электронной коммерции, давайте рассмотрим еще один уникальный плагин.

Easy Digital Downloads

Все плагины электронной коммерции, которые упоминались до сих пор, пригодны не только для физических изделий, но и для цифровых товаров и загрузок. Если же вы планируете продавать исключительно цифровые товары, вам стоит взглянуть на плагин Easy Digital Downloads (**easydigitaldownloads.com**), разработанный специально для этого. Он обладает следующими возможностями:

- загрузка фалов, доступная только авторизованным клиентам;
- возможность оплачивать сразу несколько загрузок.

Документацию о настройке Easy Digital Downloads и многое другое можно найти на веб-сайте этого продукта (**oreil.ly/cpU1C**).

Об авторах

Брайан Мессенленер начал свою карьеру разработчика программного обеспечения в Корпусе морской пехоты США в 2000 году. Он является сооснователем AlphaWeb.com, AppPresser.com и SchoolPresser.com — интернет-компаний, которые специализируются на разработке веб-приложений и мобильных решений на основе WordPress. С 2008 года Брайан работает с WordPress в качестве специалиста, разрабатывая веб-сайты и мобильные решения для таких клиентов, как журнал TIME, NBC, Microsoft, Discovery Channel, Constant Contact, Uber, Campbell's Soup, HEB, Starbucks, YMCA, государственные школы Ньюарка (Нью-Джерси), Служба национальных парков США и другие. Брайана можно найти в "Твиттере" по псевдониму **@bmess**.

Джейсон Коулман занимает должность генерального директора Stranger Studios и является ведущим разработчиком Paid Memberships Pro — платформы подписок для WordPress. Уже более пяти лет он использует WordPress в разработке приложений на PHP. Джейсон рад помочь своим клиентам в получении оплаты через Paid Memberships Pro, позволяя им открывать новые и развивать существующие предприятия. Вы можете связаться с ним на сайте **therealjasoncoleman.com** или в "Твиттере", по псевдониму **@jason_coleman**.

Предметный указатель

D

E

F

G

P

Q

R

S

T

U

V

W

X

Y

Z

А

Б

В

Н

О

П

P

C

Т

У

Ф

Х

Ц

Ш

Э

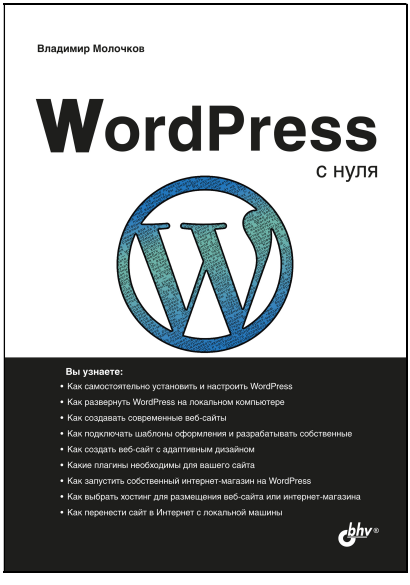
www.bhv.ru

Молочков В.
WordPress с нуля

Отдел оптовых поставок

E-mail: opt@bhv.ru

Создай свой сайт с WordPress!



- Как самостоятельно установить и настроить WordPress
- Как развернуть WordPress на локальном компьютере
- Как создавать современные веб-сайты
- Как подключать шаблоны оформления и разрабатывать собственные
- Как создать веб-сайт с адаптивным дизайном
- Какие плагины необходимы для вашего сайта
- Как запустить собственный интернет-магазин на WordPress
- Как выбрать хостинг для размещения веб-сайта или интернет-магазина
- Как перенести сайт в Интернет с локальной машины

Программа WordPress — бесплатная система управления содержимым сайта с открытым исходным кодом. С помощью WordPress можно создавать как простые веб-сайты «визитки», так и сложные проекты — блоги, корпоративные порталы, интернет-магазины.

Прочитав книгу, вы узнаете, как установить и настроить WordPress на локальном компьютере, как выбрать подходящий хостинг и перенести разработанный вами сайт в Интернет. Вы научитесь создавать сайты, адаптированные для компьютеров, ноутбуков и различных типов мобильных устройств. Книга поможет выбрать подходящий шаблон оформления или разработать собственный. Используя плагины, вы сможете превратить ваш сайт в популярный блог, корпоративный портал или интернет-магазин.

Эта книга поможет вам построить свой первый веб-сайт на WordPress без знаний программирования и навыков дизайна, даже если до этого вы никогда не занимались веб-разработкой.

Владимир Петрович Молочков, кандидат педагогических наук, преподаватель высшей квалификационной категории Политехнического колледжа Новгородского госуниверситета им. Ярослава Мудрого. Автор 18 книг по компьютерной тематике и более 200 статей в СМИ.

O'REILLY®

Разработка веб-приложений на WordPress

WordPress — это гораздо больше, чем платформа для создания блогов. Если у вас есть базовый опыт работы с PHP, HTML, CSS и JavaScript, вы можете использовать WordPress для создания быстрых, масштабируемых, безопасных и настраиваемых веб-приложений, мобильных приложений, веб-сервисов и целых сетей веб-сайтов. Наряду с основными функциями WordPress и взаимодействием с базой данных вы узнаете, как разрабатывать собственные плагины, темы и сервисы практически для любого вида веб-приложений или мобильных приложений.

В обновленном втором издании рассказывается о новых функциях и возможностях, добавленных в WordPress. Все примеры кода из книги доступны на веб-сервисе GitHub.

- Сравните WordPress с традиционными средами разработки приложений
- Используйте темы для настройки внешнего вида и плагины для обеспечения функциональности бэкенда
- Получите советы по выбору и созданию плагинов WordPress
- Регистрируйте собственные типы записей (CPT) и таксономии
- Управляйте учетными записями, ролями пользователей и доступом к данным
- Настраивайте асинхронное поведение с помощью jQuery
- Используйте WordPress для разработки мобильных приложений для iOS и Android
- Интегрируйте библиотеки PHP, внешние API и плагины веб-сервисов
- Получайте платежи с помощью плагинов eCommerce и membership
- Узнайте, как ускорить и масштабировать приложение WordPress
- Расширьте WordPress REST API и создайте собственные конечные точки (custom endpoints)
- Узнайте о разработке блоков WordPress Gutenberg

«WordPress – это не просто программное обеспечение, это движение, которое де-факто становится операционной системой Интернета. Когда вы научитесь использовать WordPress в качестве платформы для приложений, вы окажетесь на пике третьей волны его роста».

— Мэтт Малленвег, соучредитель WordPress

Брайан Мессенленер — соучредитель нескольких веб-компаний, специализирующихся на разработке пользовательских и мобильных приложений на базе WordPress. Создавал решения на базе WordPress для таких клиентов, как журнал TIME, компании NBC, Microsoft и Uber.

Джейсон Коулман помог запустить несколько стартапов, используя WordPress в качестве фреймворка для приложений. В настоящее время возглавляет разработку Paid Memberships Pro — коммерческого плагина для организации платной подписки.

ISBN 978-5-9775-6753-4

191036, Санкт-Петербург,
Гончарная ул., 20
Тел.: (812) 717-10-50,
339-54-17, 339-54-28
E-mail: mail@bhv.ru
Internet: www.bhv.ru

bhv®