Создание своих объектов в WordPress

1. Пользовательские плагины

Истинная мощь WordPress для разработчиков приложений заключается в том, что вы всегда можете создать свой собственный плагин для расширения функциональности WordPress.

Чтобы разработать плагин, сначала создайте новую папку в каталоге `wp-content/plugins` с именем `my-plugin` и поместите в эту папку файл PHP с именем `my-plugin.php`. В `my-plugin.php` вставьте код следующего примера и не стесняйтесь изменять любое из значений.

```
<?php
/*
* Plugin Name: My Plugin
* Plugin URI: https://bwawwp.com/my-plugin/
* Description: This is my plugin description.
* Author: messenlehner, strangerstudios
* Version: 1.0.0
* Author URI: https://bwawwp.com
* License: GPL-2.0+
* License URI: http://www.gnu.org/licenses/gpl-2.0.txt
*/
```

Сохраните ваш файл `my-plugin.php`. Поздравляем, вы являетесь автором плагина WordPress! Даже если ваш плагин пока еще ничего не делает, вы сможете увидеть его на странице `/wp-admin/plugins.php` и активировать его.

Давайте заставим ваш плагин на самом деле делать что-то. Мы добавим текст в нижний колонтитул вашей инсталляции WordPress. Скопируйте и сохраните код следующего примера после информации о плагине.

```
function my_plugin_wp_footer() {
    echo 'Я изучил «Создание веб-приложений с помощью WordPress»
        и теперь я гений WordPress!';
}
add_action('wp_footer', 'my_plugin_wp_footer');
```

После того как обновите страницу и перейдете на веб-сайт, вы должны заметить новое сообщение в нижнем колонтитуле. Теперь вы можете настроить этот базовый плагин так, чтобы он делал все, что вы хотите.

Разберемся, что делает этот плагин. При открытии любой страницы на вашем сайте WordPress выполняет ряд функций. Некоторые нужны для настройки отображения, другие — загружают данные, третьи — позволяют выполнить некоторый пользовательский код. Встроенная функция `add_action` — одна из таких. Она позволяет добавить что-то своё в нужный раздел сайта.

Общее название функций, позволяющих внедрить дополнительный функционал в базовый код фреймворка — **Хуки WordPress**.

2. Хуки WordPress

Хуки делают добавление функциональности в плагины и темы WordPress простым и легким. Существуют два типа хуков — **события** и **фильтры** — и они являются двумя наиболее мощными инструментами в WordPress.

События

Если хук событий (технически функция `do_action()`) существует в коде, работающем на WordPress, то вы можете вставить свой код, вызвав функцию `add_action()` и передав имя хука события и пользовательскую функцию вместе с кодом, который вы хотите выполнить:

```
do_action($tag, $arg);
```

Вот доступные аргументы:

- `$tag` — имя выполняемого хука события;
- `$arg` — один или несколько дополнительных аргументов, которые передаются в функцию, вызываемую из функции `add_action()`, ссылающейся на эту функцию `do_action()`.

Вы можете создать свой собственный хук в теме или плагине, добавив свои собственные функции `do_action()`. Однако гораздо чаще вы будете использовать готовые хуки, уже установленные в ядре WordPress или других плагинах.

Например, предположим, что вы хотели проверить, вошел ли пользователь в систему при первой загрузке WordPress, но перед отображением какого-либо вывода в браузере.

```
add_action('init', 'my_user_check');
function my_user_check() {
    if (is_user_logged_in()) {
        // сделать что-то, потому что пользователь вошел в систему
    }
}
```

В ядре WordPress есть хук событий `do_action('init')`, и мы вызываем функцию `my_user_check()` из функции `add_action()`. В любой момент времени, когда код выполняется, если он попадает в хук события `'init'`, тот запускает нашу собственную функцию `my_user_check()`, чтобы сделать все, что мы хотим, прежде чем продолжить.

Самые популярные события WordPress

Фильтры

Фильтры являются своего рода хуками событий в том смысле, что вы можете использовать их везде, где они есть в WordPress. Однако вместо того, чтобы вставлять свой собственный код, где есть хук или функция `do_action()`, вы фильтруете возвращаемое значение существующих функций, которые вызывают функцию `apply_filters()` в ядре WordPress, плагинах и/или темах.

Другими словами, с помощью фильтров вы можете захватить контент до того, как он будет вставлен в базу данных или до того, как он будет отображен в браузере в виде HTML:

```
apply_filters($tag, $value, $var);
```

- `$tag` — название хука фильтра;
- `$value` — значение, к которому может быть применен фильтр;
- `$var` — любые дополнительные переменные, такие как строка или массив, которые передаются в функцию фильтра.

Если вы выполните поиск в файлах WordPress по названию `apply_filters()`, то обнаружите, что эта функция вызывается повсеместно, и, как хуки событий, её также можно добавлять и вызывать из любой темы или плагина.

В любом месте кода, работающего на вашем сайте WordPress, где вы видите вызываемую функцию `apply_filters()`, вы можете фильтровать значение, возвращаемое этой функцией.

Для примера мы собираемся отфильтровать заголовки всех сообщений, прежде чем они будут отображены в браузере. Мы можем подключиться к любым существующим фильтрам через функцию `add_filter()` .

```
add_filter($tag, $function, $priority, $accepted_args);
```

- `$tag` — имя хука фильтра, который вы хотите фильтровать. Оно должно соответствовать параметру `$tag` вызова функции `apply_filters()` , для которого вы хотите отфильтровать результаты;
- `$function` — имя пользовательской функции, используемой для фактической фильтрации результатов;
- `$priority` — это число устанавливает приоритет, в котором будет выполняться ваша функция `add_filter()` , по сравнению с другими местами в коде, которые могут ссылаться на тот же тег хука фильтра. По умолчанию это значение равно `10` ;
- `$accepted_args` — вы можете установить количество параметров, которые может принять ваша пользовательская функция, которая обрабатывает фильтрацию. По умолчанию установлено значение `1` , что является параметром `$value` функции `apply_filters()` .

Начнем с добавления фильтра для изменения заголовка любого сообщения, возвращаемого в браузер. Мы знаем о хуке фильтра для `'the_title'` , который выглядит так:

```
apply_filters('the_title', $title, $id);
```

Здесь `$title` — это заголовок поста, а `$id` — идентификатор поста.

```
add_filter('the_title', 'my_filtered_title', 10, 2);
function my_filtered_title($value, $id) {
    $value = '[' . $value . ']';
    return $value;
}
```

Представленный в примере код программы заключает любые заголовки сообщений в скобки. Если бы заголовок вашего сообщения был **Привет, мир**, теперь он стал бы **[Привет, мир]**.

Обратите внимание, что мы не использовали аргумент `$id` в нашей пользовательской функции. Если бы мы хотели, то могли бы добавить скобки только для конкретных определенных идентификаторов постов.

Самые популярные фильтры WordPress

3. Шорткоды

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 `[myshortcode]`
2. Шорткоды с атрибутами: `[myshortcode id='1' type='text']`
3. Заключающие шорткоды, например `[myshortcode id='1']` ... некоторый контент здесь ... `[/myshortcode]`

Основной этап создания шорткодов — определение функции обратного вызова для вашего шорткода с помощью функции `add_shortcode()`. Произвольное вложенное содержимое передается обратному вызову в качестве второго параметра — `$content`.

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

```
/*
    Обратный вызов для шорткода [msg]
*/
function sp_msg_shortcode($atts, $content)
{
    $content = do_shortcode($content); // разрешаем вложенные шорткоды
    $r = '<div class="message"><p>'.$content.'</p></div>';
    return $r;
}
add_shortcode('msg', 'sp_msg_shortcode');
```

4. Пользовательские типы записей

При установке WordPress по умолчанию у вас уже есть несколько типов записей. Типы записей, которые вам наиболее знакомы, это страницы и посты, но есть еще несколько.

Стандартные типы записей

Страница

Страницы WordPress — это статические страницы контента, такие как домашняя страница, страница "О сайте", контактная информация, биографии или любая другая пользовательская страница, которую вы хотите создать. Страницы могут быть неограниченно вложены в любую иерархическую структуру.

Публикация

Ваши публикации — это ваш блог или новости, или называйте как хотите, сюда входит весь ваш контент, который индексируется поисковыми системами Интернета. Вы можете классифицировать свои посты, помечать их ключевыми словами, устанавливать даты публикации и многое другое.

Вложение

Всякий раз, когда вы загружаете изображение или файл в публикацию, WordPress сохраняет этот файл не только на сервере, но и как публикацию.

Редакции

WordPress заботится о вас и сохраняет ваши посты в качестве редакций каждый раз, когда вы или кто-либо другой редактирует их. Эта функция включена по умолчанию, и вы можете обратиться к истории, чтобы вернуть содержимое к предыдущему состоянию, если что-то напутано.

Элемент меню навигации

Каждый раз, когда вы создаете пользовательское меню с помощью основного редактора меню WordPress (**wp-admin** → **appearance** → **menus**), вы храните записи с информацией для ваших меню.

Свои типы записей

Как и стандартные типы записей WordPress, вы можете формировать свои записи для управления любыми данными, которые вам нужны, в зависимости от того, что вы создаете. Каждая запись на самом деле просто пост, используемый по-другому. Вы можете зарегистрировать записи:

- для меню ужина в ресторане,

- для автомобилей от автодилера,
- для отслеживания информации о пациенте и документах в кабинете врача,
- для почти всего, что только можно представить.

Кейс 1. Меню ужина в ресторане (Ресторанный бизнес)

Задача: Ресторану нужно динамическое меню, где блюда можно группировать по категориям (салаты, горячее, десерты, винная карта). Шеф-повар должен легко менять цену или состав, а официанты — выводить меню на планшеты.

Почему стандартный пост не подходит?
У блюда есть специфические поля: цена, вес/объём, калорийность, аллергены, время приготовления. В стандартном посте этого нет.

Что создаём:

- **Тип записи:** `dish` (Блюдо)
- **Таксономии:** `menu_category` (Категория меню: Салаты, Супы, Горячее, Десерты, Бары)
- **Произвольные поля (метаданные):**
  - `dish_price` — цена (число)
  - `dish_weight` — вес/объём (граммы/мл)
  - `dish_calories` — калорийность (опционально)
  - `dish_allergens` — аллергены (глютен, лактоза, орехи) — массив или текст
  - `dish_cooking_time` — время приготовления в минутах

Пример вывода на фронтенде:

```
Салат «Цезарь» ..... 450 руб.
(250 г, 320 ккал, 10 мин)
Состав: курица, пармезан, соус.
Аллергены: яйцо, молоко, горчица
```

Дополнительно: Можно сделать таксономию `season` (Сезонное меню: осень-2025, лето-2025).

Кейс 2. Автомобили от автодилера (Автосалон)

Задача: Автосалон выкладывает автомобили в наличии. У каждой машины — технические характеристики, цена, фото, статус (в наличии / под заказ / продан).

Что создаём:

- **Тип записи:** `car` (Автомобиль)
- **Таксономии:**
  - `car_brand` (Марка: BMW, Toyota, Kia) — иерархическая или нет
  - `car_body_type` (Тип кузова: седан, хэтчбек, кроссовер)
  - `car_transmission` (КПП: механика, автомат, робот)
- **Произвольные поля:**
  - `car_price` — цена (число)
  - `car_year` — год выпуска
  - `car_engine_volume` — объём двигателя (1.6, 2.0)
  - `car_engine_power` — мощность (л.с.)
  - `car_mileage` — пробег (км)
  - `car_color` — цвет
  - `car_status` — статус (in_stock, on_order, sold) — лучше выпадающий список
  - `car_vin` — VIN-номер (скрытое поле)

Пример URL: `/cars/bmw-x5-2023/`

Где пригодится: Вывод на карте, фильтр по цене/году/пробегу, сравнение авто.

Кейс 3. Информация о пациенте и документах (Медицинская система)

Задача: Частная клиника ведёт электронные карты пациентов. Для каждого визита нужно хранить жалобы, диагноз, назначения, результаты анализов (файлы).

Что создаём:

- **Тип записи:** `patient_visit` (Визит пациента)
- **Привязка к пользователю:** можно через `post_author` или отдельное поле `patient_id` (связь с пользователем WordPress или отдельной таблицей).
- **Таксономии:**
  - `diagnosis_category` (Категория диагноза: кардиология, неврология, травматология)
  - `doctor_department` (Отделение врача)
- **Произвольные поля:**
  - `visit_date` — дата визита
  - `complaints` — жалобы (текст)
  - `diagnosis` — диагноз (текст)
  - `prescriptions` — назначения (textarea)
  - `attachments` — вложения (ID файлов из медиатеки)

- `blood_pressure` — давление (строка: "120/80")
- `temperature` — температура

- **Вложения:** используйте `add_post_meta( $visit_id, 'prescription_file', $file_id )` для хранения ID файлов.

Важно: Медицинские данные требуют дополнительной защиты (SSL, ограничение доступа по ролям).

Кейс 4. Документооборот (внутренний портал)

Добавлю к вашему списку:

Задача: В компании нужно хранить входящие документы (письма, счета, акты) с возможностью поиска по номеру, дате, контрагенту и статусу согласования.

Что создаём:

- **Тип записи:** `incoming_document`
- **Таксономии:**
  - `doc_type` (Тип документа: счёт, договор, акт, письмо)
  - `doc_status` (Статус: черновик, на согласовании, подписан, архив) — иерархическая или нет
- **Произвольные поля:**
  - `doc_number` — входящий номер
  - `doc_date` — дата документа
  - `contractor` — контрагент
  - `responsible_id` — ID ответственного пользователя
  - `file_id` — ID загруженного PDF
  - `resolution` — резолюция (текст)

Связи: Можно связать с пользователем (`responsible_id`) через `get_userdata()` .

Как выбрать, нужен ли свой тип записи?

Задайте три вопроса:

1. **Есть ли у сущности уникальные поля?**
   Если кроме заголовка и текста нужно хранить цену, дату, автора (не автора поста), файлы — нужен свой тип записи.

2. **Будет ли много таких объектов (100+)?**
   Свои типы записей лучше работают с массовыми операциями, фильтрами в админке и REST API.
3. **Нужна ли своя логика URL / шаблонов?**
   Если вы хотите, чтобы URL был `/catalog/smartfony/apple-iphone-15/` , а не `/blog/...` — нужен свой тип.

Создание своего типа записей

В нашем примере мы собираемся создать запись для управления домашним заданием на веб-странице преподавателя. Он хочет создать какую-нибудь публикацию, где будет добавлять задания, а его студенты смогут найти их на странице группы. Преподаватель также хотел бы иметь возможность загружать дополнительные документы и публиковать комментарии на случай, если у какого-нибудь студента возникнут вопросы.

Мы можем хранить эту информацию так же, как обрабатываются посты, и отображать их для конечного пользователя в теме, используя тот же цикл, что и с постами.

Мы можем хранить эту информацию так же, как обрабатываются посты, и отображать их для конечного пользователя в теме, используя тот же цикл, что и с постами.

Функция `register_post_type($post_type, $args)`

Вы можете зарегистрировать запись с помощью функции `register_post_type()` . И в большинстве случаев вы регистрируете свою запись в файле `functions.php` вашей темы или в файле пользовательского плагина. Эта функция принимает два параметра — имя создаваемого вами поста и массив аргументов:

- `$post_type` — название вашей записи; в нашем примере имя — `homework` (домашняя работа). Эта строка не должна содержать более 20 символов и не может включать заглавные буквы, пробелы или специальные символы, кроме дефиса или нижнего подчеркивания. Если вы создаете плагин для публичного распространения, то можете использовать префикс в имени вашей записи, чтобы избежать конфликтов, на случай если какой-нибудь другой плагин имеет такое же имя;
- `$args` — это массив множества различных аргументов, которые определят настройку вашей записи. Аргументов достаточно много, их легко можно найти в литературе. Мы рассмотрим возможные аргументы на примере.

Пример регистрации типов записей `homework` и `submissions`:

```
// пользовательская функция для регистрации типа записи "homework"
function schoolpress_register_post_type_homework() {
    register_post_type('homework', [
        'labels' => [
            'name'  => 'Работы',
            'singular_name' => 'Работа'
        ],
        'public' => true,
        'has_archive' => true,
    ]);
}

// вызываем нашу пользовательскую функцию с помощью хука init
add_action('init', 'schoolpress_register_post_type_homework');


// пользовательская функция для регистрации типа записи "submissions"
function schoolpress_register_post_type_submission() {
    register_post_type('submissions', [
        'labels' => [
            'name' => 'Дополнения',
            'singular_name' => 'Дополнение'
        ],
        'public' => true,
        'has_archive' => true,
    ]);
}

// вызываем нашу пользовательскую функцию с помощью хука init
add_action('init', 'schoolpress_register_post_type_submission');
```

Если вы вставите этот код в файл `functions.php` вашей активной темы или активного плагина, то в панели администрирования WordPress под пунктом меню **Комментарии** появятся два новых пункта меню, которые называются **Работы** и **Дополнения**.

Шаблон функции регистрации типа записи с примерами

полей и комментариями

```
// 1. Регистрируем тип записи "book" (Книги)
function my_register_book_post_type() {
    register_post_type('book', [
        'labels'      => [
            'name'               => 'Книги',
            'singular_name'      => 'Книга',
            'add_new'            => 'Добавить новую',
            'add_new_item'       => 'Добавить новую книгу',
            'edit_item'          => 'Редактировать книгу',
            'new_item'           => 'Новая книга',
            'view_item'          => 'Смотреть книгу',
            'search_items'       => 'Искать книги',
            'not_found'          => 'Книг не найдено',
            'not_found_in_trash' => 'В корзине нет книг',
        ],
        'public'      => true,            // виден на фронте и в админке
        'has_archive' => true,            // архив /book/
        'menu_icon'   => 'dashicons-book', // иконка в меню
        'supports'    => [                // какие поля по умолчанию
            'title',
            'editor',
            'thumbnail',
            'excerpt',
            'custom-fields'               // чтобы добавить произвольные поля
        ],
        'show_in_rest' => true,           // поддержка Gutenberg / REST API
    ]);
}
add_action('init', 'my_register_book_post_type');

// 2. Добавляем метабоксы с дополнительными полями (цена, автор, год)
function my_add_book_metaboxes() {
    add_meta_box(
        'book_details',
        'Детали книги',
        'my_book_details_callback',
        'book',
        'normal',
        'high'
    );
}
```

```
add_action('add_meta_boxes', 'my_add_book_metaboxes');

// 3. Отображение полей внутри метабокса
function my_book_details_callback($post) {
    // получаем сохранённые значения
    $author = get_post_meta($post->ID, 'book_author', true);
    $year   = get_post_meta($post->ID, 'book_year', true);
    $price  = get_post_meta($post->ID, 'book_price', true);

    // Nonce для безопасности
    wp_nonce_field('save_book_details', 'book_details_nonce');
    ?>
    <p>
        <label>Автор:</label><br>
        <input type="text" name="book_author" value="<?php echo esc_attr($author); ?>" style="w
    </p>
    <p>
        <label>Год издания:</label><br>
        <input type="number" name="book_year" value="<?php echo esc_attr($year); ?>">
    </p>
    <p>
        <label>Цена (руб):</label><br>
        <input type="text" name="book_price" value="<?php echo esc_attr($price); ?>">
    </p>
    <?php
}

// 4. Сохраняем значения полей при сохранении записи
function my_save_book_details($post_id) {
    // проверяем nonce
    if (!isset($_POST['book_details_nonce']) ||
        !wp_verify_nonce($_POST['book_details_nonce'], 'save_book_details')) {
        return;
    }
    // защита от автосохранения
    if (defined('DOING_AUTOSAVE') && DOING_AUTOSAVE) return;
    if (!current_user_can('edit_post', $post_id)) return;

    if (isset($_POST['book_author'])) {
        update_post_meta($post_id, 'book_author', sanitize_text_field($_POST['book_author']));
    }
    if (isset($_POST['book_year'])) {
        update_post_meta($post_id, 'book_year', intval($_POST['book_year']));
```

```
    }
    if (isset($_POST['book_price'])) {
        update_post_meta($post_id, 'book_price', sanitize_text_field($_POST['book_price']));
    }
}
add_action('save_post_book', 'my_save_book_details');
```

Комментарий:

- `register_post_type` — создаёт новый тип записи.
- `add_meta_box` — добавляет блок с дополнительными полями.
- `get_post_meta` / `update_post_meta` — работа с произвольными полями.
- Всегда используйте `sanitize_text_field` и проверки прав доступа.

Если вы устали от написания функций для регистрации различных пользовательских типов записей, то можете использовать замечательный плагин под названием Custom Post Type UI.

5. Пользовательские таксономии

Мы уже упоминали таксономии, но что конкретно представляет собой таксономия — не рассматривали.

Таксономии группируют сообщения по терминам. Возьмем, к примеру, категории сообщений и теги сообщений — это просто встроенные таксономии, прикрепленные к стандартному типу записей **пост**. Вы можете определить столько пользовательских таксономий или категорий, сколько хотите, и распределить их по нескольким типам записей.

Например, мы можем создать специальную таксономию **Предмет**, которая включает в себя все учебные предметы в качестве терминов и привязана к нашей записи **Домашняя работа**.

Таксономии хороши для данных, которые служат для группировки сообщений вместе. В нашем примере в качестве таксономии имеет смысл кодировать такие вещи, как **предмет задания** (например, математика или английский язык). Мы хотим выполнять запросы типа *получить все задания по математике*. Это легко сделать с помощью запроса таксономии.

Вы можете зарегистрировать свои собственные таксономии с помощью функции `register_taxonomy()` , которая находится в файле `wp-includes/taxonomy.php` .

Практические кейсы для таксономий

Кейс 1. Фильтр товаров по бренду

Для типа записи `product` создаётся таксономия `brand` с терминами: Apple, Samsung, Xiaomi.

Кейс 2. Рубрикатор рецептов

Тип записи `recipe` + таксономия `cuisine` (русская, итальянская, японская).

Кейс 3. Город / регион для мероприятий

Тип записи `event` + таксономия `location` (Москва, СПб, Казань).

Функция `register_taxonomy($taxonomy, $object_type, $args)`

Функция принимает три параметра:

- `$taxonomy` — обязательная строка имени объекта таксономии. В нашем примере название нашей таксономии — `"subject"` .
- `$object_type` — обязательный массив или строка записей, к которым вы присоединяете эту таксономию. В нашем примере мы используем строку и присоединяем таксономию предмета к типу записи `"homework"` . Мы можем установить более одного типа записи, передав массив имен типов записей.
- `$args` — необязательный массив из множества аргументов определяет, как настроена ваша пользовательская таксономия.

```
// пользовательская функция для регистрации таксономии "Предмет"
function schoolpress_register_taxonomy_subject() {
    register_taxonomy(
        'subject',
        'homework',
        [
            'label' => "Предметы",
            'rewrite' => ['slug' => 'subject'],
            'hierarchical' => true
        ]
    );
}

// вызываем нашу пользовательскую функцию с помощью хука init
add_action('init', 'schoolpress_register_taxonomy_subject');
```

Обратите внимание, что в коде этого примера таксономия предмета настроена как вложенная категория, поскольку ее иерархический аргумент имеет значение `true`. Вы можете создать столько предметов, сколько захотите, и вкладывать их друг в друга.

Пример регистрации неиерархической таксономии (как метки)

```
// Добавляем к книгам таксономию "Жанр" (как метки, а не категории)
function my_register_genre_taxonomy() {
    register_taxonomy(
        'genre',               // имя таксономии
        'book',                // к какому типу записи привязываем
        [
            'labels' => [
                'name'          => 'Жанры',
                'singular_name' => 'Жанр',
                'add_new_item'  => 'Добавить новый жанр',
                'search_items'  => 'Искать жанры',
            ],
            'public'       => true,
            'hierarchical' => false,   // false = как метки (не вложенные)
            'show_in_rest' => true,
            'rewrite'      => ['slug' => 'genre'],
        ]
    );
}
add_action('init', 'my_register_genre_taxonomy');
```

6. Готовый плагин: Книги

Ниже представлен готовый плагин, который создаёт тип записи `book` с полями (автор, год, цена), таксономию `genre` и добавляет шорткод для вывода книг на фронтенде.

Создайте файл `my-books-plugin.php` в папке `/wp-content/plugins/my-books-plugin/` со следующим содержимым:

```php
<?php
/**
 * Plugin Name: My Books Plugin
 * Description: Добавляет тип записи "Книги", таксономию "Жанры" и шорткод для вывода книг.
 * Version: 1.0
 * Author: Ваше имя
 */

// Защита от прямого доступа
if (!defined('ABSPATH')) {
    exit;
}

// 1. Регистрация типа записи "book"
function mb_register_book_post_type() {
    register_post_type('book', [
        'labels' => [
            'name'               => 'Книги',
            'singular_name'      => 'Книга',
            'add_new'            => 'Добавить новую',
            'add_new_item'       => 'Добавить новую книгу',
            'edit_item'          => 'Редактировать книгу',
            'new_item'           => 'Новая книга',
            'view_item'          => 'Смотреть книгу',
            'search_items'       => 'Искать книги',
            'not_found'          => 'Книг не найдено',
            'not_found_in_trash' => 'В корзине нет книг',
        ],
        'public'      => true,
        'has_archive' => true,
        'menu_icon'   => 'dashicons-book',
        'supports'    => ['title', 'editor', 'thumbnail', 'excerpt'],
        'show_in_rest' => true, // поддержка Gutenberg
        'rewrite'     => ['slug' => 'books'], // URL: /books/название-книги
    ]);
}
add_action('init', 'mb_register_book_post_type');

// 2. Регистрация таксономии "Жанр" (неиерархическая, как метки)
function mb_register_genre_taxonomy() {
    register_taxonomy(
        'genre',
        'book',
```

```
        [
            'labels' => [
                'name'          => 'Жанры',
                'singular_name' => 'Жанр',
                'add_new_item'  => 'Добавить новый жанр',
                'search_items'  => 'Искать жанры',
            ],
            'public'       => true,
            'hierarchical' => false,
            'show_in_rest' => true,
            'rewrite'      => ['slug' => 'genre'],
        ]
    );
}
add_action('init', 'mb_register_genre_taxonomy');

// 3. Добавление метабоксов с дополнительными полями
function mb_add_book_metaboxes() {
    add_meta_box(
        'mb_book_details',
        'Детали книги',
        'mb_book_details_callback',
        'book',
        'normal',
        'high'
    );
}
add_action('add_meta_boxes', 'mb_add_book_metaboxes');

// 4. Отображение полей в метабоксе
function mb_book_details_callback($post) {
    $author = get_post_meta($post->ID, 'mb_book_author', true);
    $year   = get_post_meta($post->ID, 'mb_book_year', true);
    $price  = get_post_meta($post->ID, 'mb_book_price', true);

    wp_nonce_field('mb_save_book_details', 'mb_book_details_nonce');
    ?>
    <p>
        <label>Автор:</label><br>
        <input type="text" name="mb_book_author" value="<?php echo esc_attr($author); ?>" style=
    </p>
    <p>
        <label>Год издания:</label><br>
```

```
        <input type="number" name="mb_book_year" value="<?php echo esc_attr($year); ?>">
    </p>
    <p>
        <label>Цена (руб):</label><br>
        <input type="text" name="mb_book_price" value="<?php echo esc_attr($price); ?>">
    </p>
    <?php
}

// 5. Сохранение полей
function mb_save_book_details($post_id) {
    if (!isset($_POST['mb_book_details_nonce']) ||
        !wp_verify_nonce($_POST['mb_book_details_nonce'], 'mb_save_book_details')) {
        return;
    }
    if (defined('DOING_AUTOSAVE') && DOING_AUTOSAVE) return;
    if (!current_user_can('edit_post', $post_id)) return;

    if (isset($_POST['mb_book_author'])) {
        update_post_meta($post_id, 'mb_book_author', sanitize_text_field($_POST['mb_book_author
    }
    if (isset($_POST['mb_book_year'])) {
        update_post_meta($post_id, 'mb_book_year', intval($_POST['mb_book_year']));
    }
    if (isset($_POST['mb_book_price'])) {
        update_post_meta($post_id, 'mb_book_price', sanitize_text_field($_POST['mb_book_price']
    }
}
add_action('save_post_book', 'mb_save_book_details');

// 6. Шорткод для вывода списка книг
// Использование: [books_list genre="фантастика" posts_per_page="5"]
function mb_books_list_shortcode($atts) {
    $atts = shortcode_atts([
        'genre'          => '',
        'posts_per_page' => -1,
        'orderby'        => 'title',
        'order'          => 'ASC',
    ], $atts);

    $args = [
        'post_type'      => 'book',
        'posts_per_page' => intval($atts['posts_per_page']),
```

```
    'orderby'        => sanitize_text_field($atts['orderby']),
    'order'          => sanitize_text_field($atts['order']),
];

if (!empty($atts['genre'])) {
    $args['tax_query'] = [
        [
            'taxonomy' => 'genre',
            'field'    => 'slug',
            'terms'    => sanitize_text_field($atts['genre']),
        ],
    ];
}

$query = new WP_Query($args);
ob_start();

if ($query->have_posts()) {
    echo '<div class="books-list">';
    while ($query->have_posts()) {
        $query->the_post();
        $author = get_post_meta(get_the_ID(), 'mb_book_author', true);
        $year   = get_post_meta(get_the_ID(), 'mb_book_year', true);
        $price  = get_post_meta(get_the_ID(), 'mb_book_price', true);
        ?>
        <div class="book-item">
            <h3><a href="<?php the_permalink(); ?>"><?php the_title(); ?></a></h3>
            <?php if ($author): ?>
                <p><strong>Автор:</strong> <?php echo esc_html($author); ?></p>
            <?php endif; ?>
            <?php if ($year): ?>
                <p><strong>Год:</strong> <?php echo esc_html($year); ?></p>
            <?php endif; ?>
            <?php if ($price): ?>
                <p><strong>Цена:</strong> <?php echo esc_html($price); ?> руб.</p>
            <?php endif; ?>
            <p><?php the_excerpt(); ?></p>
        </div>
        <?php
    }
    echo '</div>';
    wp_reset_postdata();
} else {
```

```
        echo '<p>Книг не найдено.</p>';
    }

    return ob_get_clean();
}
add_shortcode('books_list', 'mb_books_list_shortcode');
```

После активации плагина у вас появится:

- новый тип записи **Книги** в админке;
- таксономия **Жанры**;
- шорткод `[books_list]` .

7. Вывод на фронтенде стандартными средствами темы

WordPress автоматически использует шаблоны темы для отображения произвольных типов записей.

A) Файл `single-book.php` (для отдельной книги)

Создайте в папке вашей активной темы файл `single-book.php` . Пример содержания:

```php
<?php get_header(); ?>

<div class="container">
    <?php while (have_posts()) : the_post(); ?>
        <h1><?php the_title(); ?></h1>

        <?php if (has_post_thumbnail()) : ?>
            <div class="book-cover">
                <?php the_post_thumbnail('medium'); ?>
            </div>
        <?php endif; ?>

        <div class="book-meta">
            <p><strong>Автор:</strong> <?php echo esc_html(get_post_meta(get_the_ID(), 'mb_book_
            <p><strong>Год:</strong> <?php echo esc_html(get_post_meta(get_the_ID(), 'mb_book_ye
            <p><strong>Цена:</strong> <?php echo esc_html(get_post_meta(get_the_ID(), 'mb_book_p

            <p><strong>Жанры:</strong>
                <?php
                $genres = wp_get_post_terms(get_the_ID(), 'genre');
                if (!empty($genres)) {
                    $genre_links = [];
                    foreach ($genres as $genre) {
                        $genre_links[] = '<a href="' . get_term_link($genre) . '">' . $genre->na
                    }
                    echo implode(', ', $genre_links);
                } else {
                    echo '—';
                }
                ?>
            </p>
        </div>

        <div class="book-description">
            <?php the_content(); ?>
        </div>

    <?php endwhile; ?>
</div>

<?php get_footer(); ?>
```

Б) Файл `archive-book.php` (архив всех книг)

Создайте в папке темы файл `archive-book.php` :

```
<?php get_header(); ?>

<div class="container">
    <h1>Все книги</h1>

    <?php if (have_posts()) : ?>
        <div class="books-archive">
            <?php while (have_posts()) : the_post(); ?>
                <div class="book-preview">
                    <h2><a href="<?php the_permalink(); ?>"><?php the_title(); ?></a></h2>
                    <p><strong>Автор:</strong> <?php echo esc_html(get_post_meta(get_the_ID(),
                    <p><?php the_excerpt(); ?></p>
                </div>
            <?php endwhile; ?>
        </div>

        <?php the_posts_pagination(); ?>

    <?php else : ?>
        <p>Книг пока нет.</p>
    <?php endif; ?>
</div>

<?php get_footer(); ?>
```

В) Файл `taxonomy-genre.php` (архив по жанру)

Если перейти по ссылке `/genre/fantastika/` , WordPress будет искать этот шаблон:

```php
<?php get_header(); ?>

<div class="container">
    <h1>Книги в жанре: <?php echo single_term_title(); ?></h1>

    <?php if (have_posts()) : ?>
        <div class="books-archive">
            <?php while (have_posts()) : the_post(); ?>
                <div class="book-preview">
                    <h2><a href="<?php the_permalink(); ?>"><?php the_title(); ?></a></h2>
                    <p><?php the_excerpt(); ?></p>
                </div>
            <?php endwhile; ?>
        </div>
        <?php the_posts_pagination(); ?>
    <?php else : ?>
        <p>Нет книг в этом жанре.</p>
    <?php endif; ?>
</div>

<?php get_footer(); ?>
```

8. Использование шорткода

Шорткод можно вставить:

- в любой пост или страницу (через редактор);
- в виджет "Текст";
- прямо в файл темы через `do_shortcode()` .

Примеры:

```
[books_list]
```

Выведет **все** книги, отсортированные по названию.

```
[books_list genre="fantastika" posts_per_page="5"]
```

Выведет **5** книг в жанре "фантастика".

```
[books_list orderby="date" order="DESC" posts_per_page="10"]
```

Выведет **10 последних** добавленных книг.

Вставка шорткода в файл темы:

```php
<?php echo do_shortcode('[books_list genre="fantastika" posts_per_page="3"]'); ?>
```

9. Иерархия шаблонов для произвольных типов записей

URL	Какой файл ищет тема
`/books/название-книги/`	`single-book.php` → `single.php` → `index.php`
`/books/`	`archive-book.php` → `archive.php` → `index.php`
`/genre/фантастика/`	`taxonomy-genre.php` → `taxonomy.php` → `archive.php`

Важно: После добавления новых файлов шаблонов может потребоваться сбросить постоянные ссылки (Настройки → Постоянные ссылки → Сохранить).

10. Что дальше?

- Добавьте **поля для миниатюры** — WordPress поддерживает её по умолчанию для `'thumbnail'` в `supports`.
- Добавьте **метабокс для рейтинга книги** по аналогии с автор/год.
- Создайте **виджет** "Популярные жанры" со списком терминов таксономии `genre`.