# Формы от Дмитрия Полос

В Wordpress есть свои скрипты для почты: wp-admin/admin-post.php и admin-ajax.php.

#### admin-post.php

<form class=" " id=" " action="<?php echo admin\_url( 'admin-post.php' ); ?>" method="**post**">

где admin\_url() - адрес папки wp-admin. Вместо 'admin-post.php может быть admin-ajax.php

Может еще использоваться:

<?php echo esc\_url(admin\_url( 'admin-post.php' ));?> - чтобы не было символов, не допустимых для url-адресов.

На серевере данные из формы будут в массиве $POST. Ключи в этом массиве - атрибуты name. В Wordpress обязательно нужен ключ action. В **форме** добавить:

<input type="hidden" name="**action**" value="**rl-modal-form**">

где rl-modal-form - префикс для хука

В admin-post.php выбирается - на какой хук подписаться в зависимости от того - авторизован пользователь или нет. Если нужно для всех, то подписаться сразу на 2 хука. В **functions.php** добавляем:

add\_action( 'admin\_post\_nopriv\_rl-modal-form', 'rl\_modal\_form\_handler' );

add\_action( 'admin\_post\_rl-modal-form', 'rl\_modal\_form\_handler' );

function rl\_modal\_form\_handler() {

echo 'все получили';

var\_dump( $\_POST );

header('Location: ' . home\_url()); // переадресация на главную страницу

}

#### admin-ajax.php. Лайки

Будем пользоваться Local Storage

Как отправить данные? Вставить в элемент (кнопку) атрибут

**data-href**="<?php echo esc\_url(admin\_url( 'admin-ajax.php' )); ?>"

или объявить переменную

<script>

**window.ajaxUrl** = <?php echo esc\_url(admin\_url( 'admin-ajax.php' )); ?>

</script>

начало скрипта - обработки лайков:

<script>

window.addEventListener(**'load'**, function() {

const likeBtn = document.querySelector('.like');

**try** {

if( !**localStorage.getItem('liked')** ) {

**localStorage.setItem('liked', '');**

}

} **catch** (e) {

console.log(e);

}

let haslike = false;

});

</script>

Для отправки данных **в скрипте (в верстке):**

const data = new FormData();

data.append('action', **'post-likes'**);

где formData.append(name, value) – добавляет к объекту поле с именем name и значением value

В **functions.php**

add\_action( **'admin\_ajax\_nopriv\_post-like'**, 'rl\_likes' );

add\_action( **'admin\_ajax\_post-like'**, 'rl\_likes' );

function rl\_likes() {

echo 'все получили';

**wp\_die();//** остановка работы wordpress

}

Неполный код фронтенда:   
 <button

class="like"

data-href="<?php echo esc\_url(admin\_url( 'admin-ajax.php' )); ?>"

>

<p>Нравится</p>

<div class="like\_\_count">0</div>

</button>

<script>

window.addEventListener('load', function() {

const likeBtn = document.querySelector('.like');

try {

if( !localStorage.getItem('liked') ) {

localStorage.setItem('liked', '');

}

} catch (e) {

console.log(e);

}

let haslike = false;

let todo = '';

likeBtn.addEventListener('click', () => {

if(!haslike) {

haslike = true;

likeBtn.style.backgroundColor = 'red';

todo = 'plus';

} else {

haslike = false;

likeBtn.style.backgroundColor = '#fff';

todo = 'minus';

}

});

const data = new FormData();

data.append('action', 'post-like');

data.append('todo', todo);

const xhr = new XMLHttpRequest();

xhr.open('POST', likeBtn.getAttribute('data-href'));

xhr.send(data);

likeBtn.disable = true;

xhr.addEventListener('readystatechange', function() {

if(xhr.readyState !==4) return;

if(xhr.status === 200) {

likeBtn.querySelector('.like\_\_count').innerText = xhr.responseText;

let lovalData = localStorage.getItem('liked');

if(haslike){

localStorage.setItem('liked', '+');

} else {

localStorage.setItem('liked', '-');

}

} else {

console.log(xhr.statusText);

}

likeBtn.disable = false;

});

});

</script>

#### Колонки в админке

Неполный код Добавляем колонку для лайков в админке

add\_action( 'manage\_posts\_custom\_column', 'rl\_like\_column', 5, 2 );

function rl\_like\_column(){ }

add\_filter( 'manage\_posts\_columns', 'rl\_add\_col\_likes' );

#### admin-post.php. Бэкенд. Создание своего типа записи Заявки, который не выводится через фронтенд

Сделаем обработку форму на бэкенде. Сделаем **новый тип записи Заявки**, который не будет отображаться на фронтенде, не будет иметь своего url, но будет доступен в админке администратору. Через ACF будут созданы только те поля, которые нужно редактировать - комментарий по заявке и статус заявки. Другие поля - имя, email, дата, id записи.

На кнопку Записаться вешаем дата атрибут для записи id страницы.

<a href="#form2" **data-id**="<?php echo **$id**; ?>">

<button class="btn">Записаться</button>

</a>

Через ja этот атрибут будет передан в **форму** в скрытое поле

<input type="**hidden**" class="" name="**form-post-id**" value="**2**">

В **functions.php** создадим новый тип записи:

// Регистрируем новый тип записей - Заявки

register\_post\_type( **'orders'**, [

'labels' => [

'name' => \_\_('Заявки', 'robot\_lending'),

'singular\_name' => \_\_('Заявка', 'robot\_lending'),

'add\_new' => \_\_('Добавить нового', 'robot\_lending'),

'add\_new\_item' => \_\_('Добавить новую заявку', 'robot\_lending'),

'edit\_item' => \_\_('Редактировать заявку', 'robot\_lending'),

'new\_item' => \_\_('Новая заявка', 'robot\_lending'),

'view\_item' => \_\_('Смотреть заявку', 'robot\_lending'),

'search\_items' => \_\_('Искать заявку', 'robot\_lending'),

'not\_found' => \_\_('Не найдено', 'robot\_lending'),

'not\_found\_in\_trash' => \_\_('Не найдено в корзине', 'robot\_lending'),

'parent\_item\_colon' => '',

'menu\_name' => \_\_('Заявки', 'robot\_lending'),

],

'public' => false,

'show\_ui' => true, //Определяет нужно ли создавать логику управления типом записи из админ-панели.

'show\_in\_menu' => true,//Показывать ли тип записи в администраторском меню и где именно показывать управление типом записи. Аргумент show\_ui должен быть включен!

'menu\_position' => 22,

'menu\_icon' => 'dashicons-format-chat',

'hierarchical' => false,

'supports' => ['title'],

'has\_archive' => false

]);

Оставим **Заголовок** для редактирования. Остальные поля вставим сами

В **functions.php**

Подписываемся на хук

add\_meta\_boxes хук-событие . WP 3 - Позволяет зарегистрировать метаблок для любого типа записи. Срабатывает после того, как все встроенные метаблоки были добавлены.

add\_meta\_box() WP 2.5.0 - Добавляет дополнительные блоки (meta box) на страницы редактирования/создания постов, постоянных страниц или произвольных типов записей в админ-панели.

add\_action( 'add\_meta\_boxes', 'rl\_meta\_boxes');

function rl\_meta\_boxes() {

add\_meta\_box(

'rl\_order\_date',

'Дата заявки',

'rl\_order\_date\_cb', // функция, которая отвечает за верстку поля

'orders'// к каким типам записей добавлять

);

}

function rl\_order\_date\_cb( $post\_obj ) {

$date = get\_post\_meta( $post\_obj->ID, 'rl\_order\_date', true );

$date = $date ? $date : '';

echo '<span>' . $date. '</span>';

}

После этого в админке будут добавлены поля без возможности их редактирования.

Если полей много, то, чтобы не повторять код, можно его оптимизировать:

function rl\_meta\_boxes() {

$fields = [

'rl\_order\_date' => 'Дата заявки',

'rl\_order\_name' => 'Имя клиента',

'rl\_order\_email' => 'email',

'rl\_order\_choice' => 'Выбор клиента' // id

];

foreach ($fields as $slug => $text) {

# code...

add\_meta\_box(

$slug,

$text,

'rl\_order\_fields\_cb', // функция, которая отвечает за верстку поля

'orders',// к каким типам записей добавлять

'advanced',// куда вставлять - выбрали по умолчанию

'default',// приоритет

$slug // дополнительные аргументы для rl\_order\_fields\_cb

);

}

}

// уже дописанная функция

function rl\_order\_fields\_cb( $post\_obj, $slug) {

$slug = $slug['args'];

$data = '';

switch ($slug) {

case 'rl\_order\_date':

$data = $post\_obj->post\_date;

break;

case 'rl\_order\_choice':

$id = **get\_post\_meta**( $post\_obj->ID, $slug, true );

$title = **get\_the\_title**( $id );

$data = 'Клиент выбрал <strong>' . $title. '</strong><br>';

//$type = **get\_post\_type\_obj**(**get\_post\_type**( $id ))->labels->singular\_name;// название типа записи

//$data = 'Клиент выбрал <strong>' . $title. '</strong><br> Из раздела <strong>'. $type. '</strong>';

break;

default:

$data = **get\_post\_meta**( $post\_obj->ID, $slug, true );

$data = $data ? $data : 'Нет данных';

break;

}

echo '<p>' . $data. '</p>';

}

Поля **Статус** и **Комментарии** добавим через **Админку**. Создадим новую группу полей - **Поля для Заявок**. Отображать группу полей, если тип Записи равно Заявка.

Добавим поле **Статус Заявки** с типом Выбор (select). Варианты:

new : Новая Заявка

done : Заявка обработана

processing : Требуется уточнение

Значение по умолчанию new : Новая Заявка

Возвращаемый формат Оба

Добавим поле **Комментарий**. Тип поля - Область текста. Условная логика: Статус Заявки равно Требует уточнения

В **functions.php** дописываем функцию обработки из первого пункта

function **rl\_modal\_form\_handler()** {

$name = $\_POST['user2'] ? $\_POST['user2'] : 'Аноним';

$email = $\_POST['email2'] ? $\_POST['email2'] : false;

$choice = $\_POST['form-post-id'] ? $\_POST['form-post-id'] : 'empty';

// сохраняем только, если есть email

if( $email ) {

$name = wp\_strip\_all\_tags( $name ); //Удаляет все HTML теги из строки

$email = wp\_strip\_all\_tags( $email );

$choice = wp\_strip\_all\_tags( $choice);

//wp\_slas - Добавляет слэши (\), экранирует строку или массив строк (в том числе строки всех вложенных массивов).

//wp\_insert\_post - Безопасно вставляет/обновляет запись в базе данных. Возвращает id новой записи или 0 или объект ошибки - зависит от настроек

$id = wp\_insert\_post( wp\_slash( [

'post\_title' => 'Заявка №',

'post\_type' => 'orders',

'post\_status' => 'publish',

'meta\_input' => [

'rl\_order\_name' => $name,

'rl\_order\_email' => $email,

'rl\_order\_choice' => $choice,

]

] ));

if( $id !== 0 ){

wp\_update\_post( [

'ID' => $id,

'post\_title' => 'Заявка №' . $id

] );

// далее добавляем поля из ACF

update\_field( 'orders\_status', 'new', $id );

// Здесь еще можно вызывать wp\_mail();

}

}

header('Location: ' . home\_url()); // переадресация на главную страницу

}

Можно еще выводить столбцы в админке Статус заявки с комментариями и делать их разными цветами. Можно при этом использовать admin\_enqueue\_scripts для своих стилей и скриптов в админпанели.

#### Contact Form 7 и Flamingo

<https://contactform7.com/> - документация по плагину

Аналогично, как у Артема Исламова. Если верстка формы сломалась, придется залезать в верстку или искать фильтры. При этом нужно сохранить стили Wordpress, иначе, функциональность может быть нарушена.

Добавим тегу <**form>** свои классы и id. Для этого нужно добавит в шорткод атрибут **html\_class и html\_id**

Далее, также, как у Артема Исламова устанавливаем плагин Flamingo