链滴

【开发笔记】开发一个基于 wordpress 和 woocommerceRest 的接口插件

作者：yf98

原文链接：https://ld246.com/article/1664011967788

来源网站：链滴

近期使用ReactNative开发项目，后台用的woocommerce那一套，需要写Rest接口对接，虽然woo带的有,问题都是服务端的，客户端调用用只读密钥还行，写就没办法了，只能自己开坑写服务端的Res接口插件了。

目前插件已完成所需功能涉及：wp hook，wp rest，wp auth，woo hook，woo rest 等。基本的token认证到注册自定义产品类型，hook已有产品定制化流程，拦截rest数据进行二次定制等。

1.wordpress插件结构

下文以booking这个插件为例，记录一下开发要注意的地方，首先看一下插件结构。

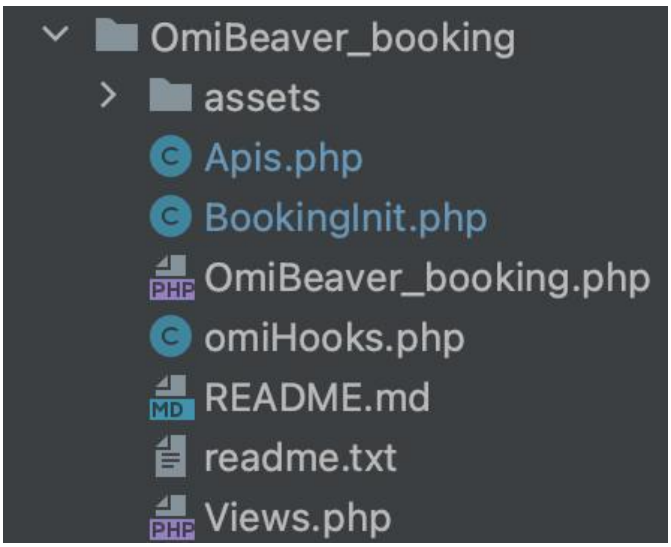
首先你需要创建一个文件夹，名称就是你的插件名称，其次，你需要创建一个同名的PHP入口文件：miBeaver_booking.php（后面会列出内容），如果你的插件不进行市场上架发布，那么只需要这一必须的文件即可，如果需要上架，那么你需要多一个必须文件：readme.txt 用来在wordpress市场发布使用。其他文件则可以根据你的项目需要自行创建，我这里是开发一个预约系统的Rest插件，其涉及，登录授权，后台面板的定制功能，这里分了

1.Apis.php（对外Rest接口）

2.BookingInit.php(程序主类)

3.omiHooks.php(插件使用的Hook聚合类)

4.Views(定制后台用到的HTML代码)



先从入口文件看

```
<?php

/**
 * Plugin Name:       OmiBeaverBooking
 * Plugin URI:        https://codecanyon.net/user/omibeaver/portfolio
 * Description:       Virtual goods course reservations Rest API
 * Version:           1.0.0
 * Requires at least: 5.2
 * Requires PHP:      7.2
 * Author:            omiBeaver
 * Author URI:        https://codecanyon.net/user/omibeaver/portfolio
```

```
 * License:           GPL v2 or later
 * License URI:       https://www.gnu.org/licenses/gpl-2.0.html
 * Update URI:        https://codecanyon.net/user/omibeaver/portfolio
 * Text Domain:       OmiBeaverBooking
 * Domain Path:       /languages
 */


require_once 'BookingInit.php';
$MACRO = new BookingInit;
```

可以看到Plugin name这里是与文件同名的，这里是需要与插件名称一直，其他元信息用来展示在插管理面板，如下所示:

TIPS：这里一定要注意，头部注释一定要和插件名称文件夹一致，否则压缩打包后，安装插件会无法别。

第一次接触wp开发的小伙伴可能一下子很难接受wp的开发模式，你会发现有很多插件都是PHP与HTL混合编程，代码里充斥着很多：

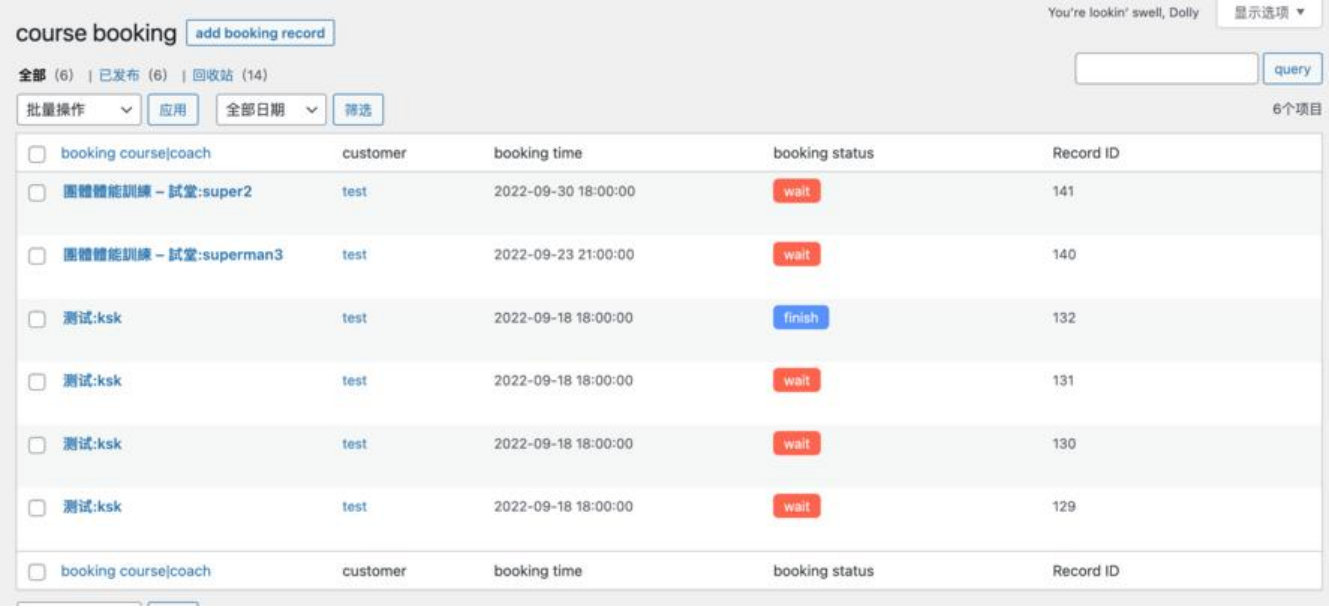
```
<?php ?> <div></div> <?php ?>
```

了解hook后，就觉得很正常了，hook就像前端写React或者Vue里的生命周期，或者安卓的activity在（wp）系统的启动后触发一系列的回调或者预定义的方法，wp内置了非常多的hook回调，你可以系统初始化的时候执行你的代码，在系统CURD的每个操作前后甚至中途插入你的代码，而你只需要用chestnut1：

```
  add_filter('manage_macro_booking_record_posts_columns', function ($columns) {
            return omiHooks::booking_admin_columns($columns);
        });

// omiHooks::booking_admin_columns
 public static function booking_admin_column($column, $post_id)
    {
        if ($column == 'booking_status') {
            Macro_views::booking_status_change($post_id);
        }
        if ($column == 'booking_time') {
            echo str_replace('T', ' ', get_post_meta($post_id, 'booking_time', true));
        }
        if($column == 'ID'){
            echo $post_id;
        }

    }


//Macro_views::booking_status_change

public static function booking_status_change($post_id)
```

```
    {
        $bg = get_post_meta($post_id, 'booking_status', true) == '0' ? 'tomato' : '#578fff';
        echo "<span onclick='omiJS.changeBookingStatus($post_id)'  style='background:$bg;padding:4px 10px;cursor: pointer;border-radius: 5px;color: #fff'>" . (get_post_meta($post_id, 'booking_status', true) == '0' ? 'wait' : 'finish') . '</span>';
    }
```

示例代码修改了帖子类型在后台表格上的展示效果，这里是展示用户预约记录的表，所以需要展示用的预约状态，我注入了一个状态变化的HTML文字，另外从数据库提取的自定义字段：预约时间在展的时候进行了一些格式化拆分。



上述eg有一些全局方法与hook稍后会说到，仅作为开头的示例展示。

1.wordpress自定义post

有了一个hook案例，可以发现可以根据系统提供的各种hook来定制自己的wp，wp常见的hook分为种

一种是actionHook，一种是filterHook，第一种是事件触发的时候，你可以注入你的自定义代码，他如何执行的呢，举个chestnut2：假如有个hook在某个帖
保存前需要执行你的自定义代码: save_post 这个hook，使用action的hook你需要使用add_action()法，第一个参数是需要使用的hook，第二个是hook的回调，第三个是优先级，第四个是参数个数。
如我们想在预约记录编辑后可以自定义预约时间或者状态，在回调里，我们可以写下如下：

```
 add_action('save_post', function ($post_id, $view) {
        omiHooks::booking_cus_save($post_id, $view);
    }, 10, 2);
//omiHooks::booking_cus_save
 public static function booking_cus_save($post_id, $view)
    {
        if ($view->post_type == 'macro_booking_record') {
            if (isset($_POST['booking_time']) && $_POST['booking_time'] != '') {
                update_post_meta($post_id, 'booking_time', $_POST['booking_time']);
            }
            if (isset($_POST['booking_status']) && $_POST['booking_status'] != '') {
```

```
            update_post_meta($post_id, 'booking_status', $_POST['booking_status']);
        }


    }
}
```

在Wp中，当系统收到post数据后，处理完正常逻辑后会在提交前执行指定位置预设的回调

```
//wp 保存post逻辑xxxxx

//wp保存前最后一步
do_action(your_callback,$some_params);
//wp 保存提交完成
```

可想而知我们在系统HTML页面上的时候，如果需要定制页面，那么只需要使用HTML页面里预设的hook即可，所以会出现php与HTML混合情况。

默认的帖子结构是不支持我们预约表单的，我们需要定制一下自己的字段与表单类型。

1.创建自定义的帖子类型

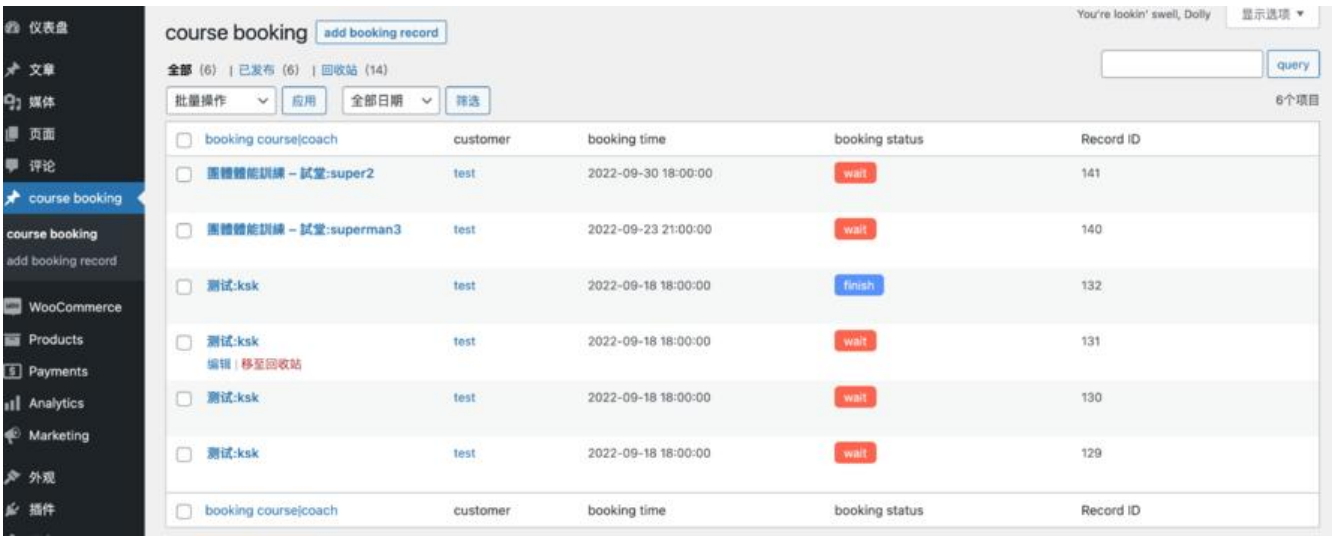
很多时候，默认的post类型不支持我们的业务，通常来说，都是倾向于写作的的字段，而预约的产品能有次数限制，有变体选择。如果我们使用woocommerce插件这里就直接定制好了。但是如果我们买了产品课程后续需要预约使用，预约记录就需要我们自己定制了。

```
add_action('init', function () {
        omiHooks::booking_record_fun();
    });
//omiHooks::booking_record_fun
public static function booking_record_fun()
  {
    register_post_type('macro_booking_record',
        array(
            'label' => 'booking record',
            'labels' => array(
                'name' => 'course booking',
                'singular_name' => 'course booking list',
                'add_new' => 'add booking record',
                'add_new_item' => 'add booking record',
                'edit' => 'update booking record',
                'edit_item' => 'update',
                'new_item' => 'create',
                'view' => 'detail',
                'view_item' => 'to detail',
                'search_items' => 'query',
                'not_found' => 'not found',
                'not_found_in_trash' => 'not found'
            ),
            'show_ui' => true,
            'show_in_menu' => true,
            'public' => true,
            'description' => 'Booking Manage',
            'has_archive' => false,
```

```
            'show_in_rest' => false,
            'supports' => [
                'title',
                'author'
            ]
        )
    );
}
```

这里注册了一个新的post类型，预约类型。参数请参阅：wp文档直达

注册以后：



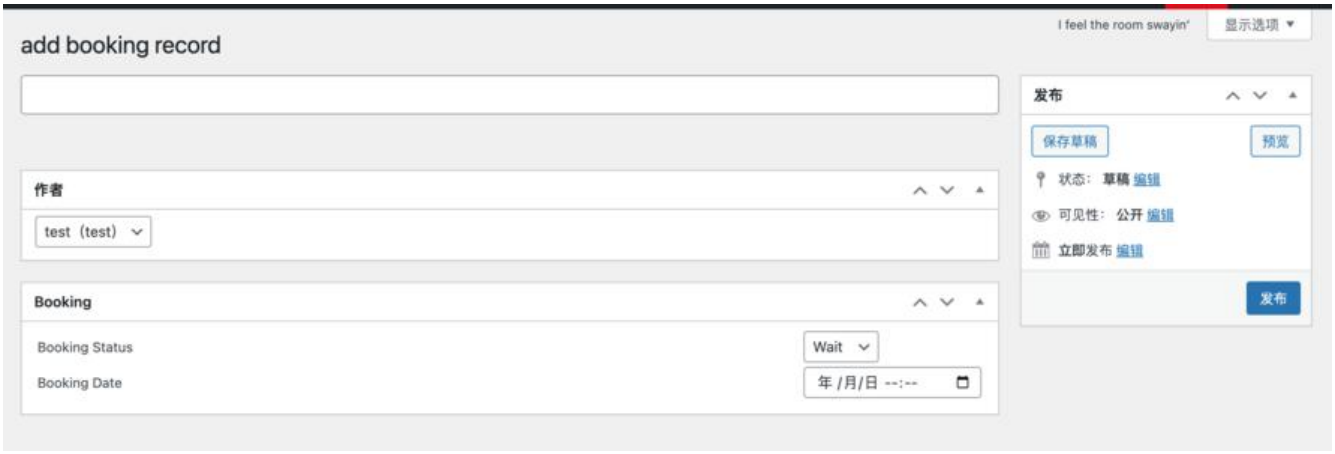
正常是没有后面几个字段的，这里需要我们的自定义字段啦，请看下面！

自定义帖子类型字段

这里需要用到Meta数据，wp预设留下了可扩展的meta类型，可以在该字段里写入新的key=>value顶级默认是数组类型。后续取出可以使用全局方法来指定key取出，无需使用array[0]格式。根据我们约需要自定义两个字段：booking_time,booking_status,下面是在后台表单里hook增加我们的字段

```
add_action('admin_init', function () {
        omiHooks::booking_view_ext();
    });

//omiHooks::booking_view_ext
public static function booking_view_ext()
    {
        add_meta_box('macro_review_meta_box', 'Booking',
            function ($view) {
                Macro_views::booking_cus_view($view);
            },
            'macro_booking_record', 'normal', 'high'
        );

    }

// Macro_views::booking_cus_view
```

```
public static function booking_cus_view($view)
   {
      ?>
      <table>
         <tr>
            <td style="width: 100%">Booking Status</td>
            <td>
               <select name="booking_status">
                  <option value="0" <?php echo $view->booking_status == '0' ? 'selected' : '' ?
>Wait</option>
                  <option value="1" <?php echo $view->booking_status == '1' ? 'selected' : '' ?
>Finish</option>
               </select>
            </td>
         </tr>
         <tr>
            <td style="width: 100%">Booking Date</td>
            <td>
               <input name="booking_time" value="<?php echo $view->booking_time ?>" typ
="datetime-local"/>

            </td>
         </tr>

      </table>
      <?php

   }
```



来读代码啦，首先这里使用了一个系统事件的hook，后台初始化的时候，执行一下我们的回调，我向系统注册了一个新的post类型。

```
add_action('init', function () {
         omiHooks::booking_record_fun();
      });
```

接下来我们需要：

1.在编辑表单支持新的字段

booking_cus_view方法，这是我们自定义的回调，这里我们拿到了当前表格的行数据，并且可以在定row的单元格进行定制输出，这里使用了select和时间选择器（wp有默认样式），在表单name里入新的字段名称即可，后面你可以使用保存的hook来存储这个自定义表单，如果还记得开始的hestnut2，会发现：

update_post_meta(post_id, 'booking_time', _POST['booking_time']);

这里就是保存的hook，因为我们使用了自定义的字段，所以需要使用meta函数来存储我们的数据。此save完成，接下来我们完成展示。

2.在后台面板展示出来

chestnut 1里面使用了filter hook：manage_macro_booking_record_posts_columns，注意hook的使用规范，如果是自定义post相关的hook，你需要在hook加上你的自定义类型名称，比如本案的

macro_booking_record

在这个hook的回调我们可以对表格的输出进行控制，隐藏或者临时新增。具体看hestnut1。

2.后台表格面板自定义交互事件

在上面的面板里有个状态显示，我们如果想通过后台来更改，打开编辑过于麻烦，直接点击这个按钮发比较合适，这里就涉及到如何绑定一个JS点击事件以及触发一个内部的ajax事件。

1.自定义后台HTML绑定JS事件（载入JS文件）

如果我们按照HTML规则来写，在：

```
 echo "<span onclick='omiJS.changeBookingStatus($post_id)'  style='background:$bg;paddin
:4px 10px;cursor: pointer;border-radius: 5px;color: #fff'>" . (get_post_meta($post_id, 'booking
status', true) == '0' ? 'wait' : 'finish') . '</span>';
```

这里触发事件的JS怎么加载的呢，起初我是直接在次数写了一个script标签来实现js代码，结果由于这的hook会根据row的渲染重复数次，又修改成全局只存在一个实例，总觉得不妥，最后按照wp的规实现的载入外部文件。

```
add_action( 'admin_enqueue_scripts', function (){
        omiHooks::loadJs();
    } );
```

上面这个hook，可以在后台初始化的时候载入我们的JS代码。

```
public static function loadJs()
  {

    wp_enqueue_script('ajax-script', plugins_url('/assets/utils.js', __FILE__));
    wp_localize_script(
      'ajax-script',
      'winter_ajax_obj',
      array(
```

```
            'ajax_url' => admin_url('admin-ajax.php'),
            'nonce' => wp_create_nonce('omiBeaver'),
        )
    );
}
```

上面的代码我们可以发现几个要注意的点，首先enqueue载入我们的代码文件后，还需要使用localiz实现本地化，不是语言的本地化，是向后台进行JS的配置，如基础URL与随机数（防止跨站）,当然也以在特定页面加载只需判断当前页面是否是你要加载的post type类型再执行即可。

看一下这个utils文件

```
/**
 * * @author  omibeaver
 * Admin Booking Manage pane JS
 * @type {{changeBookingStatus(*): void}}
 */
const omiJS = {
    changeBookingStatus(post_id){
        if(winter_ajax_obj){
            let formData = new FormData();
            formData.append('_ajax_nonce',winter_ajax_obj.nonce);
            formData.append('action',"change_booking_status");
            formData.append('post_id',post_id)
            fetch(winter_ajax_obj.ajax_url, {
                    method:'POST',
                    body:formData
                }
            ).then((body)=>{
                body.json().then((data)=>{
                    alert(data.msg);
                    setTimeout(()=>{location.reload()},1000)
                }).catch((e)=>alert(e.msg))
            })
        }else{
            alert('init failed')
        }

    }


}
```

可以看见在ajax的参数部分加入了随机数。如果只是对某个页面修改建议在后台判断页面。我这里是的全局一个工具对象。

3.wordpress Hook开发

到此，完成了新的post类型与表单定制，表格定制输出。那么代码写在哪呢，我在哪里写入我的hoo呢。

wp提供了几个常用的全局生命周期方法，插件激活，插件禁用，在入口文件里，还记得吗与文件夹件名称同名的那个文件，在这里会被wp执行，可以在这里写入hook，或者和我一样，自定义一个类始化注入全局hook即可，生命周期方法不是必须!

入口文件：

bookingInit类

```
<?php

require_once 'Views.php';
require_once 'Apis.php';
require_once 'omiHooks.php';

/**
 * @author  omibeaver
 * @name BookingInit WP init hooks
 *
 */
class BookingInit
{


    function __construct()
    {

        //create booking post type.
        add_action('init', function () {
            omiHooks::booking_record_fun();
        });

        //init routes.
        add_action('rest_api_init', function () {
            self::register_api();
        });

        //custom admin booking pane  btn column.
        add_action('admin_init', function () {
            omiHooks::booking_view_ext();
        });

        //custom admin booking pane  save post.
        add_action('save_post', function ($post_id, $view) {
            omiHooks::booking_cus_save($post_id, $view);
        }, 10, 2);


        //custom admin booking pane column.
        add_action('manage_macro_booking_record_posts_custom_column', function ($column,
post_id) {
            omiHooks::booking_admin_column($column, $post_id);
        }, 10, 2);

        //custom admin booking pane columns.
        add_filter('manage_macro_booking_record_posts_columns', function ($columns) {
            return omiHooks::booking_admin_columns($columns);
        });
```

```
        //load custom JS.
        add_action( 'admin_enqueue_scripts', function (){
            omiHooks::loadJs();
        } );

        //hook admin pane booking statue change.
        add_action('wp_ajax_change_booking_status', function () {
            omiHooks::change_booking_status();
        });

        //hook admin pane booking statue change.
        add_action('wp_ajax_nopriv_change_booking_status',function (){
            omiHooks::change_booking_status();
        });

        //custom admin booking pane row
        add_action( 'post_row_actions',function ($actions,$post ){
           return omiHooks::removeRowBtn($actions,$post );
        } ,10,2);

        //custom woocommerce product
        add_filter( 'woocommerce_rest_prepare_shop_order_object', function ($data, $post, $con
ext){
            return omiHooks::customOrderQuery($data, $post, $context);
        }, 12, 3 );

        //close auto-update tips.
        omiHooks::closeUpdate();

    }


    //start register all hooks.
    private static function register_api()
    {

        //rest loginIn
        register_rest_route('macro', '/booking_signIn', array(
            'methods' => 'GET',
            'callback' => function ($request) {
                return (new Apis($request))->bookingSignIn();
            },
            'permission_callback' => '__return_true'
        ));

        //rest query booking list by user
        register_rest_route('macro', '/booking_list', array(
            'methods' => 'GET',
            'callback' => function ($request) {
                return (new Apis($request))->bookingListQuery();
            },
            'permission_callback' => '__return_true'
        ));
```

```
        //rest signUp
        register_rest_route('macro', '/booking_signUp', array(
            'methods' => 'POST',
            'callback' => function ($request) {
                return (new Apis($request))->bookingSignUp();
            },
            'permission_callback' => '__return_true'
        ));

        //rest create new booking
        register_rest_route('macro', '/booking_create', array(
            'methods' => 'POST',
            'callback' => function ($request) {
                return (new Apis($request))->bookingCreate();
            },
            'permission_callback' => '__return_true'
        ));


        //rest change  booking status
        register_rest_route('macro', '/booking_update', array(
            'methods' => 'PUT',
            'callback' => function ($request) {
                return (new Apis($request))->bookingStatusUpdate();
            },
            'permission_callback' => '__return_true'
        ));


        register_rest_route('macro', '/create_order', array(
            'methods' => 'POST',
            'callback' => function ($request) {
                return (new Apis($request))->createOrder();
            },
            'permission_callback' => '__return_true'
        ));


    }


}
```

Views

```
<?php
```

```
/**
 * @author  omibeaver
 * View Templates
 */
class Macro_views
{

    public static function booking_cus_view($view)
    {
        ?>
        <table>
            <tr>
                <td style="width: 100%">Booking Status</td>
                <td>
                    <select name="booking_status">
                        <option value="0" <?php echo $view->booking_status == '0' ? 'selected' : '' ?
>Wait</option>
                        <option value="1" <?php echo $view->booking_status == '1' ? 'selected' : '' ?
>Finish</option>
                    </select>
                </td>
            </tr>
            <tr>
                <td style="width: 100%">Booking Date</td>
                <td>
                    <input name="booking_time" value="<?php echo $view->booking_time ?>" typ
="datetime-local"/>

                </td>
            </tr>

        </table>
        <?php

    }

    public static function booking_status_change($post_id)
    {
        $bg = get_post_meta($post_id, 'booking_status', true) == '0' ? 'tomato' : '#578fff';
        echo "<span onclick='omiJS.changeBookingStatus($post_id)'  style='background:$bg;pa
ding:4px 10px;cursor: pointer;border-radius: 5px;color: #fff'>" . (get_post_meta($post_id, 'boo
ing_status', true) == '0' ? 'wait' : 'finish') . '</span>';
    }

}
```

wordpress Rest 接口开发

Apis类：后面我们看一下代码

```
<?php
require_once 'omiHooks.php';
```

```php
/**
 * @author  omibeaver
 * Rest APIs
 */
class Apis
{

    private $request;
    private $user;
    private const WOO_SECRET = 'cs_xxx';//可写secret
    private const WOO_KEY = 'ck_xxx';//可写key
    private const REMOTE_URL = 'xxx';


    function __construct($request)
    {

        $this->request = $request;
        $is_login = wp_validate_auth_cookie($request->get_param('token'), 'macro');
        if ($is_login) {
            $userAuthInfo = wp_parse_auth_cookie($request->get_param('token'), 'macro');
            $this->user = get_user_by('login', $userAuthInfo['username'])->data;
        }
    }


    public function bookingStatusUpdate(): array
    {
        if (!$this->user) return ['code' => -1, 'msg' => 'token invalid', 'data' => null];
        $post_id = $this->request->get_param('post_id');
        $user_id = self::getUserByCookie($this->request->get_param('token'))->ID;
        if (!$post_id) {
            return ['code' => -1, 'msg' => 'params error', 'data' => null];
        }
        $args = array(
            'post_type' => 'macro_booking_record',
            'posts_per_page' => 10,
            'p' => $post_id
        );
        $data = (new WP_Query($args))->posts;
        if (count($data) != 1) {
            return ['code' => -1, 'msg' => 'content not found', 'data' => null];
        }
        if ($data[0]->post_author != $user_id) {
            return ['code' => -1, 'msg' => 'not permission', 'data' => null];
        }
        if (get_post_meta($post_id, 'booking_status', true)['booking_status'] == '0') {
            update_post_meta($post_id, 'booking_status', 1);
            return ['code' => 1, 'msg' => 'success', 'data' => null];
        } else {
            return ['code' => -1, 'msg' => 'had changed', 'data' => null];

        }
```

```
    }


    public function bookingCreate(): array
    {


        if (!$this->user) return ['code' => -1, 'msg' => 'login invalid', 'data' => null];
        $post_name = $this->request->get_param('booking_name');
        $booking_time = $this->request->get_param('booking_time');
        if (empty($booking_time) || empty($post_name)) return ['code' => -1, 'msg' => 'params
nvalid', 'data' => null];
        if (!date_create($booking_time)) return ['code' => -1, 'msg' => 'date invalid', 'data' => nu
l];
        if (strtotime($booking_time) < time()) return ['code' => -1, 'msg' => 'Can\'t make an ap
ointment before', 'data' => null];
        if (strlen($post_name) > 50) return ['code' => -1, 'msg' => 'params error', 'data' => null];
        $booking_course_title = explode(':', $post_name);
        if (count($booking_course_title) != 2) {
            return ['code' => -1, 'msg' => 'title format invalid', 'data' => null];
        }

        try {

            $booking_course_title = $booking_course_title[0];
            $booking_course_id = $this->request->get_param('course_id');

            $user_orders = (new WC_Order($booking_course_id))->get_items();

            if (count($user_orders) != 1) {
                return ['code' => -1, 'msg' => 'course not found', 'data' => null];
            }


            $user_orders = current($user_orders);
            $meta_data = current($user_orders->get_meta_data());
            $user_all_booking_count = (int)$meta_data->value;
            //Check the available schedule of the course


        } catch (Exception $exception) {
            return ['code' => -1, 'msg' => $exception->getMessage(), 'data' => null];
        }

        $args = array(
            'post_type' => 'macro_booking_record',
            'posts_per_page' => 10,
            'post_status' => 'publish',
            'author' => $this->user->ID,
            'meta_query' => [
                'booking_id' => $booking_course_id
            ]
        );
```

```
        $user_booking_count = (new WP_Query($args))->post_count;
        if ($user_booking_count > $user_all_booking_count+10) return ['code' => -1, 'msg' => '
he number of appointments has been used up', 'data' => null];
        $res = wp_insert_post([
            'post_author' => $this->user->ID,
            'post_title' => $post_name,
            'post_status' => 'publish',
            'post_name' => $post_name,
            'post_type' => 'macro_booking_record'

        ]);

        update_post_meta($res, 'booking_status', 0);
        update_post_meta($res, 'booking_time', $booking_time);
        update_post_meta($res, 'booking_course_id', $booking_course_id);
        update_post_meta($res, 'booking_course_title', $booking_course_title);
        return ['code' => 1, 'data' => ['booking_id' => $res, 'left' => $user_all_booking_count -
user_booking_count], 'msg' => 'SUCCESS'];
    }

    public function bookingSignUp(): array
    {

        $user_name = sanitize_user($this->request->get_body_params()['user_name'] ?? '');
        $password = trim($this->request->get_body_params()['password'] ?? '');
        $user_email = trim($this->request->get_body_params()['user_email'] ?? '');

        if (!$user_name || !$password || !$user_email) {
            return ['code' => -1, 'msg' => 'params error', 'data' => null];
        }

        if (strlen($user_name) > 20) {
            return ['code' => -1, 'msg' => 'params error', 'data' => null];
        }

        if (!is_email($user_email)) {
            return ['code' => -1, 'msg' => 'email error', 'data' => null];
        }

        $user_id = username_exists($user_name);
        if (!$user_id && !email_exists($user_email)) {
            $user_id = wp_create_user($user_name, $password, $user_email);
            return ['code' => 1, 'msg' => 'SUCCESS', 'data' => ['user_id' => $user_id]];
        } else {

            return ['code' => -1, 'msg' => 'account exist', 'data' => null];
        }

    }

    private static function getUserByCookie($cookie)
    {
        $userAuthInfo = wp_parse_auth_cookie($cookie, 'macro');
        return get_user_by('login', $userAuthInfo['username']);
```

```
    }


    public function bookingListQuery(): array
    {
        if (!$this->user) return ['code' => -1, 'msg' => 'login invalid', 'data' => null];
        if (empty($this->request->get_param('start_booking_time')) || empty($this->request->ge
_param('end_booking_time'))) return ['code' => -1, 'msg' => 'params booking_time  invalid', '
ata' => null];
        if (!date_create($this->request->get_param('start_booking_time')) || !date_create($this->
equest->get_param('end_booking_time'))) return ['code' => -1, 'msg' => 'params booking_ti
e  invalid', 'data' => null];
        $start_booking_time = date_create($this->request->get_param('start_booking_time'));
        $end_booking_time = date_create($this->request->get_param('end_booking_time'));
        $args = array(
            'post_type' => 'macro_booking_record',
            'posts_per_page' => 10,
            'author' => $this->user->ID,
            'meta_query' => [
                'booking_time' =>
                    array(
                        array('key' => 'booking_time', 'value' => $start_booking_time->format('Y/m/d')
 'compare' => '>=', 'type' => 'DATE'),
                        array('key' => 'booking_time', 'value' => $end_booking_time->format('Y/m/d'),
'compare' => '<=', 'type' => 'DATE'),
                    )

            ]
        );

        $data = (new WP_Query($args))->posts;
        foreach ($data as $post) {
            $post->booking_status = get_post_meta($post->ID, 'booking_status', true);
            $post->booking_time = get_post_meta($post->ID, 'booking_time', true);

        }
        return ['code' => 1, 'msg' => 'SUCCESS', 'data' => $data];


    }

    public function bookingSignIn(): array
    {

        $username = sanitize_user($this->request->get_param('username'));
        $password = trim($this->request->get_param('password'));
        $user = wp_authenticate($username, $password);
        return ['code' => 1, 'msg' => 'success', 'user' => $user, 'token' => wp_generate_auth_co
kie($user->ID, time() + 720000, 'macro')];

    }
```

```
public function createOrder(): array
{
    //默认订单数据
    $data = [
        'meta_data' => array(array(
            'key' => 'pay_status',
            'value' => '50%'
        )),

        'payment_method' => 'bacs',
        'payment_method_title' => 'Direct Bank Transfer',
        'set_paid' => true,
        'billing' => [
            'first_name' => 'testUser',
            'last_name' => 'testUser',
            'address_1' => '969 Market',
            'address_2' => '',
            'city' => 'San Francisco',
            'state' => 'CA',
            'postcode' => '94103',
            'country' => 'US',
            'email' => 'testUser@test.com',
            'phone' => '(555) 555-5555'
        ],
        'shipping' => [
            'first_name' => 'John',
            'last_name' => 'Doe',
            'address_1' => '969 Market',
            'address_2' => '',
            'city' => 'San Francisco',
            'state' => 'CA',
            'postcode' => '94103',
            'country' => 'US'
        ],
        'line_items' => [

            [
                'product_id' => 65,
                'variation_id' => 70,
                'quantity' => 1
            ]
        ],
        'shipping_lines' => [
            [
                'method_id' => 'flat_rate',
                'method_title' => 'Flat Rate',
                'total' => '0'
            ]
        ]
    ];

    try {
```

```
            $data = wp_remote_post(self::REMOTE_URL ."/wp-json/wc/v3/orders?consumer_key="
. self::WOO_KEY . "&consumer_secret=" . self::WOO_SECRET,
                array(
                    'headers' => array('Content-Type' => 'application/json'),
                    'timeout' => 30,
                    'body' => json_encode($data),
                )
            );

        } catch (Exception $exception) {
            return ['code' => -1, 'msg' => 'SUCCESS', 'data' => $exception->getMessage()];
        }

        return ['code' => 1, 'msg' => 'SUCCESS', 'data' => $data];

    }


}
```

Api接口wp默认有提供，我们有自己逻辑需要定义，所以这里使用的自定义接口，首先我们使用一个hook来初始化路由：

```
//init routes.
        add_action('rest_api_init', function () {
            self::register_api();
        });

//start register all hooks.
    private static function register_api()
    {

        //rest loginIn
        register_rest_route('macro', '/booking_signIn', array(
            'methods' => 'GET',
            'callback' => function ($request) {
                return (new Apis($request))->bookingSignIn();
            },
            'permission_callback' => '__return_true'
        ));

        //rest query booking list by user
        register_rest_route('macro', '/booking_list', array(
            'methods' => 'GET',
            'callback' => function ($request) {
                return (new Apis($request))->bookingListQuery();
            },
            'permission_callback' => '__return_true'
        ));

        //rest signUp
        register_rest_route('macro', '/booking_signUp', array(
            'methods' => 'POST',
            'callback' => function ($request) {
```

```
            return (new Apis($request))->bookingSignUp();
        },
        'permission_callback' => '__return_true'
    ));

    //rest create new booking
    register_rest_route('macro', '/booking_create', array(
        'methods' => 'POST',
        'callback' => function ($request) {
            return (new Apis($request))->bookingCreate();
        },
        'permission_callback' => '__return_true'
    ));


    //rest change  booking status
    register_rest_route('macro', '/booking_update', array(
        'methods' => 'PUT',
        'callback' => function ($request) {
            return (new Apis($request))->bookingStatusUpdate();
        },
        'permission_callback' => '__return_true'
    ));


    register_rest_route('macro', '/create_order', array(
        'methods' => 'POST',
        'callback' => function ($request) {
            return (new Apis($request))->createOrder();
        },
        'permission_callback' => '__return_true'
    ));


}
```

使用全局函数：register_rest_route来注册自定义的路由，这里需要注意生成的链接格式为：

http(s)://your_host_url/wp-json/自定义前缀/路由

API代码里面基本上很清楚，就不过多介绍了，里面使用的全局方法可以在wp文档直接找到。

woocommerce 定制接口

最后关于woocommerce定制的事情，这里我只演示一个接口作为开始吧，其他同理。

woo在wp基础上进行了深度定制，提供了非常多的hook，你可以在这里找到：woocommerce hooks

这里是提一下filter hook和action的区别，这个hook主要使用在数据获取中，下面这个
hestnut，使用woo的数据过滤器hook对订单查询记录进行定制，我们这里给订单新增了一个字段：
ooking_left，可预约次数。

```
add_filter( 'woocommerce_rest_prepare_shop_order_object', function ($data, $post, $context)
```

```
        return omiHooks::customOrderQuery($data, $post, $context);
    }, 12, 3 );

public static function customOrderQuery($data, $post, $context): array
  {

    $data->data['booking_left'] = 2;

    return $data;
  }
```

woocommerceRest调用下单

有时候会发现woo的功能比较分散，比如下订单，如果使用内部方法流程很多，这个时候可以使用Res接口直接下单：

```
public function createOrder(): array
  {
    //默认订单数据
    $data = [
      'meta_data' => array(array(
        'key' => 'pay_status',
        'value' => '50%'
      )),

      'payment_method' => 'bacs',
      'payment_method_title' => 'Direct Bank Transfer',
      'set_paid' => true,
      'billing' => [
        'first_name' => 'testUser',
        'last_name' => 'testUser',
        'address_1' => '969 Market',
        'address_2' => '',
        'city' => 'San Francisco',
        'state' => 'CA',
        'postcode' => '94103',
        'country' => 'US',
        'email' => 'testUser@test.com',
        'phone' => '(555) 555-5555'
      ],
      'shipping' => [
        'first_name' => 'John',
        'last_name' => 'Doe',
        'address_1' => '969 Market',
        'address_2' => '',
        'city' => 'San Francisco',
        'state' => 'CA',
        'postcode' => '94103',
        'country' => 'US'
      ],
      'line_items' => [

        [
```

```
                'product_id' => 65,
                'variation_id' => 70,
                'quantity' => 1
            ]
        ],
        'shipping_lines' => [
            [
                'method_id' => 'flat_rate',
                'method_title' => 'Flat Rate',
                'total' => '0'
            ]
        ]
    ];

    try {

        $data = wp_remote_post(self::REMOTE_URL ."/wp-json/wc/v3/orders?consumer_key="
. self::WOO_KEY . "&consumer_secret=" . self::WOO_SECRET,
            array(
                'headers' => array('Content-Type' => 'application/json'),
                'timeout' => 30,
                'body' => json_encode($data),
            )
        );

    } catch (Exception $exception) {
        return ['code' => -1, 'msg' => 'SUCCESS', 'data' => $exception->getMessage()];
    }

    return ['code' => 1, 'msg' => 'SUCCESS', 'data' => $data];

}
```

woocommerce和wp一样，你可以自定义下单流程的HTML内容，当然你也可以注入JS开发，上文其有使用一个哦，发现了吗！甚至你可以使用React来开发。

有其他疑问可以留言，我有空的时候会修正文章和回复。

本文涉及的插件 OmiBeaverBooking已开源：OmiBeaverBooking

yum 有定制wordpress插件需求的可联系 winter_986@qq.com