اصول اولیه برنامه نویسی افزونه وردپرس برای تولید یک محصول استاندارد

THE BASICS OF WORDPRESS PLUGIN DEVELOPMENT TO PRODUCE A STANDARD PRODUCT

نویسنده : مهرشاد درزی

توسعه دهنده وردپرس [Wordpress Developer]

وب سایت : RealWp.net

با توجه به محبوبیت زیاد وردپرس در ایجاد وب سایت ، یکی از پر درآمد ترین مشاغل می تواند توسعه و **برنامه نویسی افزونه وردپرس** برای علاقه مندان به این صنعت باشد.اما برای ورود به بازار پر رقابت ، بین برنامه نویسان قدرتمند در مارکت های خارجی مانند وب سایت Evanto می بایست در مراحل پلاگین نویسی وردپرس اصول و قواعدی را رعایت کنید.در این مقاله به برخی از این نکات اشاره خواهم کرد.

یک وردپرس جدید نصب کرده و سپس شروع به برنامه نویسی افزونه وردپرس کنید!

اولین نکته ای که می بایست در برنامه نویسی افزونه وردپرس رعایت کرد" **میزان عدم وابستگی کدها** " به دیگر افزونه یا قالب ها در وردپرس هست.زمانی که شما در یک وردپرس که در آن افزونه های زیادی نصب شده اند یا قالبی نصب شده است کار می کنید، احتمال اینکه برخی عوامل روی افزونه شما تاثیر بگذارد بسیار زیاد است.

به عنوان مثال افزونه ای را برای یکی از شرکت های ایرانی تولید کردم ، در بخشی از این افزونه تاریخ ثبت نام کاربران را می بایست به شمسی نشان می داد.از آنجا که من افزونه پارسی دیت را نصب کرده بودم اصلا متوجه این قضیه نبودم ، کاربری که قرار است این افزونه را در وردپرس خود نصب کند شاید افزونه پارسی دیت را نداشته باشد و در نتیجه افزونه من در خیلی از وب سایت ها با خطا مواجه شد.

افزونه ی شما در درجه ی اول نباید به هیچ افزونه یا قالبی در مارکت وابسته باشد زیرا ما نمی دانیم در وب سایت وردپرس کاربران چه خبر هست ، آن ها چه افزونه یا قالب هایی را نصب کرده اند.مگر زمانی که ذاتا برای توسعه یک افزونه مثلا ووکامرس ، محصولی را تولید کرده باشیم.

تابع هایی که در مستندات وردپرس هست را بر توابع خود ترجیح دهیم

برای نوشتن هر خط کد در برنامه نویسی افزونه وردپرس ، ابتدا باید بررسی کنید این امکان یا تابع قبلا در وردپرس وجود دارد یا خیر ، به عنوان مثال برای چک کردن صحیح بودن ایمیل کاربران در مستندات وردپرس تابعی به نام is_email وجود دارد.شما به هیچ وجه از تابعی غیر از آن استفاده نکنید و خودتان یک تابع جدید در افزونه خود برای صحت درستی ایمیل ایجاد نکنید.

در یکی از پروژه هایم که می بایست طبق یک دستوری فایل فشرده بک اپ سایت را باز میکرد ، ساعت ها وقت گزاشتم و کلاس های مختلف در زمینه کار با zip را در php کار کردم اما دریغ از یک سرچ ساده تا متوجه شوم که خوده وردپرس تابعی دارد به نام unzip_file که همین کار را انجام می دهد.

سوال اینجاست اصلا دلیل این کار چیست که حتما از توابع وردپرس استفاده شود ؟

دلیل اول این است که با این کار شما توسعه افزونه خود را برای توسعه دهندگان دیگر ساده می کنید. شما نسخه ابتدایی افزونه ووکامرس را به یاد ندارید اما من خوب به یاد دارم.اولین شخصی که ووکامرس را در ورژن یک تولید کرد مطمئنا در این حد امکانات خوب و متنوع نداشت. اما گروه Automattic یک کار را با دقت انجام داده بود و آن چیزی نبود جز این که طبق استاندارد وردپرس و مستندات وردپرس برنامه نویسی افزونه Woocommerce را انجام داد . در نیتجه این عمل باعث شد که راحت تر توسعه دهندگان دیگر با کدهای افزونه ارتباط برقرار کرده و براحتی آن را توسعه دهند.

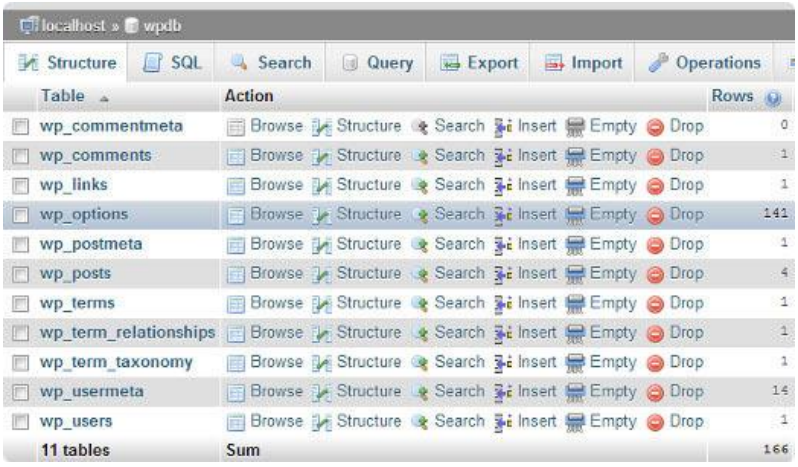
دلیل دوم هم اینکه اگر تغییراتی برای تابع مربوطه در ساختار زبان برنامه نویسی php ایجاد شد دیگر ما به زحمت نمی افتیم ، خوده وردپرس با بروز رسانی هسته ی جدید این مشکلات را برطرف کرده و

از طرفی چون ما از تابع موجود در هسته ی وردپرس استفاده کردیم ، در نیتجه مشکل ما هم برطرف می شود.

تا زمانی که نیاز ما توسط جدول های بانک اطلاعاتی وردپرس حل می شود هیچ جدولی ایجاد نکنیم

طبیعتا یکی از نیاز های برنامه نویس ، ارتباط با بانک اطلاعاتی در افزونه هست.شما به عنوان برنامه نویس ابتدا می بایست شناخت کافی از تمامی جداولی که به صورت پیش فرض در بانک اطلاعاتی هست داشته باشید و تنها در صورتی که نیاز افزونه و کدنویسی شما در چارچوب این جداول برطرف نشد ، اقدام به ایجاد یک جدول در زمان نصب افزونه کنید.

فرض کنید می خواهید یک عدد در بانک اطلاعاتی ذخیره کنید . چقدر مسخره می شود که بجای استفاده از جدول wp_option خودتان یک جدول اضافه کنید و مقادیر خود را در آن قرار دهید. اگر به افزونه ووکامرس نگاهی بیاندازید این افزونه برای قرار دادن محصولات از همان جدول Post ها در وردپرس استفاده کرده و زمانی که برای ذخیره داده های سبد خرید و کد های تخفیف نیاز خود را در جداول وردپرس ندید آن را ایجاد کرد.



همیشه زبان انگلیسی را مبنا قرار دهید

قانون" **توسعه پذیر بودن افزونه** "که یادتان هست.این اصل هم تاکیدی بر همین قانون دارد.زبان اصلی وردپرس در حالت پیش فرض United State هست. پس چه بهتر همیشه افزونه خود را به زبان en

یا همان انگلیسی برنامه نویسی کنیم و در صورتی که نیاز داشتیم زبان های دیگر مانند فارسی به آن اضافه کنیم ، از طریق تابع Textdomain این عمل را انجام دهیم.برای این کار می توانید به صورت زیر عمل کنید:

```
1. add_action( 'plugins_loaded', 'myplugin_load_textdomain' );
2. function myplugin_load_textdomain() {
3. load_plugin_textdomain( 'my-plugin', false, basename( dirname(
   __FILE__ ) ) . '/languages' );
4. }
```

قانون توسعه پذیر بودن افزونه : افزونه ی شما می بایست به شکلی نگارش شود که هر توسعه دهنده وردپرس در هر کجای دنیا با نصب یک وردپرس ساده مسیر شما را ادامه و امکاناتی به آن کم یا زیاد کند.

هر تابع یا فایل تنها در مسیری که به آن نیاز هست بارگزاری شود

من سال ها وقت خودم رو به تحقیق در مورد استارت آپ های بزرگ مبتنی بر **هسته وردپرس** صرف کردم.یکی از بزرگترین عامل برای موفقیت یک وب سایت وردپرسی ، سرعت بارگزاری آن است. شما می بایست برنامه نویسی افزونه وردپرس را به شکلی انجام دهید که **هر کد یا فایل تا تابع تنها در جایی که نیاز هست بارگزاری شود** تا بدین شکل پردازش سرور درگیر پروسس های بدون عملکرد نشود. اکثر افزونه ها این مشکل را دارند که تمامی فایل های آن در هر پروسس بارگزاری می شود ، بدون آن که به آن نیازی باشد.

اگر تازه کار هستید ابتدا یک کاغذ و قلم بردارید و این چارت رو در ذهن خود پیاده سازی کنید: تمامی پروسس های وردپرس در حالت عادی شامل ٤ پروسه هست : **بخش نمایش وب سایت** ، **بخش مدیریت** ، **بخش Admin ajax** و بخش **Rest Api** .شما می توانید در فایل php اصلی افزونه خودتان ابتدا توابعی که مربوط به هر کدام هست را توسط دستورات شرطی جدا کنید و به هر کدام اجازه بارگزاری در جای خودش را بدهید.

به عنوان مثال هیچ class یا function که مربوط به نمایش تنظیمات در مدیریت وردپرس هست نباید در وب سایت برای کاربران بارگزاری شود پس من میتوانم به صورت زیر عمل کنم:

```
1. if ( is_admin() ) {
2. include ('admin_page_plugin.php');
3. }
```

در مثال بالا فقط زمانی که در مدیریت هستیم توابع آن بارگزاری می شود.یا مثلا این مورد را نگاه کنید

```
1. global $pagenow;
2. if( is_admin() and ($pagenow =="edit-tags.php" ||
   $pagenow=="term.php")) { require_once 'class/Tax-meta-
   class/core.php';  }
```

کد بالا را در یکی از افزونه های خود ایجاد کردم.این کد تنها زمانی که در مدیریت هستیم و داخل یکی از صفحات مدیریت دسته بندی یا برچسب ها وردپرس هستیم بارگزاری خواهد شد.

هیچ وقت کوئری های اضافه تولید نکنید

این اصل ن تنها در وردپرس بلکه در هر زبان برنامه نویسی وجود دارد که برای دریافت اطلاعات از بانک اطلاعاتی می بایست طوری چیدمان داشت که با کم ترین کوئری به اطلاعات دسترسی پیدا کرد. گاها برنامه نویسان فقط می نویسند و نتیجه را مد نظر قرار می دهند. اما اصلا دقت نمی کنند که می تواند درابعاد بزرگ چقدر کوئری ایجاد شود و سرعت پردازش سایت خیلی زیاد شود.

برای این که کوئری های ایجاد شده در وردپرس را دیباگ کنید. ابتدا کد زیر را در فایل wp-config.php قرار داده:

```
1. define('SAVEQUERIES', true);
```

این کد به وردپرس ، فرمان می دهد هر کوئری که توسط کلاس wpdb اجرا می شود را در متغیر ذخیره کند. حال می توانید در فوتر فایل افزونه خود این کد را ایجاد کنید تا لیست تمامی کوئری ها را مشاهده کنید . فقط یادتان نرود پس از پایان کار کد را حذف کنید:

```
1.global $wpdb;
2. echo "<pre>";
3. print_r($wpdb->queries);
4. echo "</pre>";
```

برای رویداد های مهم افزونه از Action و Filter در وردپرس استفاده کنید

این مسئله هم تابع قانون توسعه پذیر بودن افزونه هست.شما چه زمانی میتوانید به یک برنامه نویس این اجازه را دهید که افزونه شما را توسعه دهد ؟ در حالتی که از سیستم Hook در وردپرس استفاده کنید. به عنوان مثال فرض کنید افزونه ای تولید کردید ، در یک تابع آن مبلغی را به عنوان کارمزد از مبلغ نهایی کم میکند این تابع را میبایست به شکل زیر بنویسید:

```
1. function tax($number) {
2. $tax = 100;
3. $tax = apply_filters( 'change_tax_number', $tax );
4. $number = $number - $tax;
5. return $number;
6. }
```

من تابعی نوشتم که از هر مبلغ پایانی ۱۰۰ تومان به عنوان کارمزد کم میکند و عدد نهایی را خروجی میگیرد اما برای متغیر tax یک فیلتر ایجاد کردم که توسعه دهندگان دیگر بتوانند طبق شرایط خود آن را تغییر دهند.

مثلا من میخواهم در وب سایت خودم به کاربرانی که نویسنده سایت من هستند هیچ مبلغی به عنوان مالیات گرفته نشود ، پس می نویسم:

```
add_filter('change_tax_number', function($tax){
/* Check User is Author*/
if( is_user_logged_in() && is_author(get_current_user_id()) ) {
    $tax = 0;
}
return $tax;
});
```

اگر افزونه شما نیاز به پکیج یا قابلیت خاصی دارد در همان مرحله ی نصب آن را چک کنید

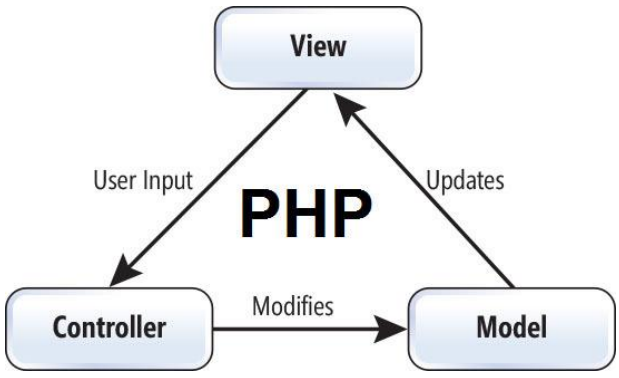
زمانی که افزونه شرکت پارسا اسپیس را در وردپرس واقعی می نوشتم ، این افزونه نیاز به اتصال به سیستم **Api**این شرکت داشت و تقریبا ۸۰ درصد توابع حول دستور **Curl**می چرخید. پس این خیلی منطقی بود که من در همان مرحله ی نصب چک کنم که آیا تابع Curl در سرور فعال هست یا خیر . و اگر فعال نبود اجازه نصب افزونه داده نشود و کاربر با پیغام خطا مواجه شود.این خیلی بهتر از این است که کاربر افزونه را نصب کند و ببینید افزونه کار نمی کند و خطا می دهد و در نتیجه باعث نارضایتی او شود.

برای اجرای دستورات و توابع در لحظه نصب افزونه از register_activation_hook در وردپرس استفاده کنید.

همیشه برنامه نویسی MVC را بر ساختار معمولی ترجیح دهید

نمیدانم چرا همه ی برنامه نویسان php زمانی که پروژه ای را خودشان می نویسند یا مثلا در فریم ورک لاراول انجام می دهند ، همه را در ساختار MVC پیاده سازی می کنند .ولی زمانی که پای وردپرس می نشینند اصلا یادشان می رود که میتوان شی گرایی هم کار کرد.

جدا از این که چقدر سرعت پردازش افزونه با استفاده از Composer و autoload بهتر و بالاتر می رود مشکل هم نام بودن تابع هم برایمان پیش نمی آید.



فرض کنید تابعی نوشتید در افزونه برای چک کردن ایمیل با نام . wp_is_email شما از کجا میدانید که شاید کاربر ، افزونه یا قالبی نصب نکرده باشد که دقیقا یک تابع با این نام در آن وجود داشه باشد و در نهایت وردپرس به ارور برخورد کند.

البته میتوانید همانند خیلی از افزونه ها ، نام اختصاری در ابتدای تابع بگذارید مثلا افزونه ووکامرس اکثر تابع های آن با عبارت _wc آغاز می شود یا افزونه acf هم با عبارت._afc

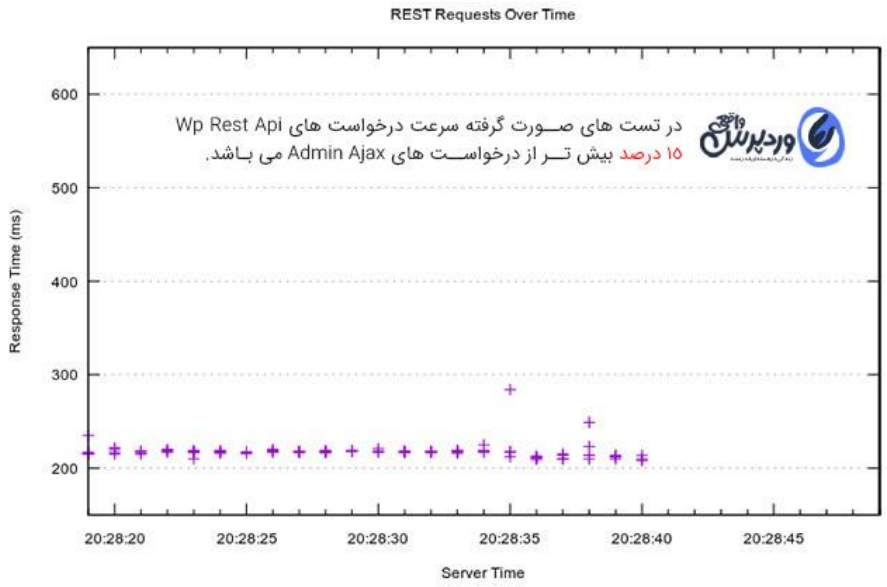
اگر قدرت برنامه نویسی افزونه را بصورت MVC ندارید ، پیشنهاد میکنم از پکیج Wppb برای شروع یک پروژه برنامه نویسی افزونه استفاده کنید.

مراقب سرعت پردازش و بارگزاری و امنیت در درخواست های Ajax باشید !

دو نکته کلیدی همواره در درخواست **Ajax** مطرح هست:

سرعت Ajax در وردپرس

در تست های به عمل آمده توسط متخصصین وردپرس ، عملکرد درخواست هایی که به Rest Api وردپرس داده شده ، ۱۶ درصد سریع تر از درخواست هایی بوده که از طریق Admin-ajax وردپرس دریافت شده.به همین دلیل هست که بسیـــاری از افزونه های مطرح مانند ACF یا Yoast Seo به **WP Rest Api** کوچ کردند.



پیشنهاد من هم استفاده از Rest Api بجای Admin ajax هست.البته برای درخواست های کم کوچ کردن به Rest Api را ترجیح و اولویت قرار ندهید.

یکی از کارهایی که افزونه ووکامرس انجام داده مسیر درخواست های Ajax را خودش جداگانه با پارامتر Wc-ajaxایجاد کرده است.شما هم میتوانید اگر خیلی درخواست های فراوانی در برنامه نویسی افزونه وردپرس خود دارید، مسیر های جداگانه تعریف کنید تا با بیشترین سرعت درخواست های شما بارگزاری شود.

بهبود امنیت در درخواست Ajax

به جرات میتوان گفت بیش از ۵۰ درصد دلیل نفوذ به وردپرس از طریق اشتباهاتی بوده که برنامه نویسان افزونه و قالب در درخواست های آجاکس ، شامل درخواست POST یا GET مرتکب شده اند. اگر هر خط کدی که نوشتید را سه بار میخوانید و چک می کنید.هر خط کد که در Ajax وردپرس ایجاد می کنید را ده بار بخوانید.

۱- حتما کد امنیتی یا **nonce** را در ابتدا برای اعتبار سنجی اعمال کنید.
۲- اگر درخواست شما شامل یکسری ورودی هایی است ، تمامی ورودی ها را چک کنید و در صورت عدم وجود حتی یک مورد ، کاربر را در php بصورت کامل exit کنید.
۳- در نوع ورودی ها دقت کنید ، اگر قرار هست نام خانوادگی شخص را دریافت کنید ولی تنها یک عدد برای این مقدار بازگشته اعتبار سنجی کامل انجام و خطا را نشان دهید.

تا حد امکان فراخوانی فایل CSS و JavaScript را در صفحه های مورد نیاز انجام دهید

اگر در حال نوشتن افزونه ای هستید که برای یک صفحه خاص ، امکانی را مهیا می کند. کاملا منطقی است که به وردپرس دستور دهید تنها در همان صفحه ، فایل های Css و Js مرتبط با افرونه شما را بارگزاری کند .روش های بسیار متداولی وجود دارد ، مثلا در یک فیلد از کاربر بپرسید ، در کدام صفحه میخواهید این امکان نمایش داده شود.حال که شناسه یا ID صفحه را دریافت کردید با دستورات شرطی مثل is_page آن فایل را بارگزاری کنید.

در داخل مدیریت نیز می توانید از طریق مختلف مانند دریافت اطلاعات **Screen**صفحه متوجه شوید ، که در چه صفحه ای می بایست فایل های Css و Js را فراخوانی کنید مثلا من در افزونه ای فقط در صفحه خودش فایل font Awesome را به این صورت فراخوانی کردم:

```
1. $screen = get_current_screen();
2. /*
3. Load Only Plugin Page
4. */
5. if( $screen->id =='toplevel_page_my_plugin') {
6. wp_enqueue_style( 'Font-Awesome', plugin_dir_url( __FILE__ ) .
   'css/font-awesome/font-awesome.min.css', array(), $this->version,
   'all' );
7. }
```

سخن پایانی

همیشه به ما گفته اند که رقیبان خود را زیر نظر داشته باشید و همواره آنالیز و بررسی کنید و مسیری که آنها رفته اند را دنبال کنید تا راه موفقیت را پیدا کنید.مسلما قواعد های پایه ی زیادی برای توسعه و برنامه نویسی افزونه وردپرس وجود دارد.

من هم به شما پیشنهاد میکنم هیچ معلمی بالا تر از آن نیست که فایل های افزونه های معروف را با ویرایشگر باز کنید و خط به خط آن را بررسی کنید و پاسخ تمامی سوالاتتان را در اینترنت و از اساتید سوال کنید تا بتوانید بهترین محصول را تولید کنی.

کم افرادی نبودند که تنها با یک قالب یا افزونه نیاز مالی چندین سال خود را برطرف کرده اند.
از کجا معلوم نفر بعدی شما نباشید ؟

برای مشاهده مقالات در زمینه آموزش وردپرس از وب سایت "وردپرس واقعی" دیدن فرمائید.

WWW.REALWP.NET