استفاده بهینه از کلاس WP Query در قالب وردپرس

OPTIMAZE WP_QUERY CLASS IN WORDPRESS THEME

نویسنده : مهرشاد درزی

توسعه دهنده وردپرس [Wordpress Developer]

وب سایت : RealWp.net

یکی از موردی که اغلب **سرعت پایین وردپرس** را منجر می شود استفاده نادرست از کلاس Wp_Query می باشد.همانطور که می دانید بخش اصلی یک وب سایت وردپرسی محتوای آن هست که در دو جدول Post و PostMeta نگه داری می شوند.ما بوسیله کلاس Wp Query با ارسال درخواست هایی به بانک اطلاعاتی این محتوا را دریافت و در صفحه وب سایت منتشر می کنیم.

استفاده مستقیم از کلاس WP Query در **قالب وردپرس** مثل آن می ماند برای یک نفر که با یک پرس غذا سیر می شود هزینه ی زیادی انجام دهیم و ۱۰ پرس سفارش دهیم.آخر چه می شود ؟ خوب مشخص هست پول زیادی را هدر دادیم و خیلی از غذاها را دور ریختیم.

آیا فکر نمی کنید همین کار دقیقا در وب سایت وردپرس شما رخ می دهد ، اما با این تفاوت که ما در آنجا از منابع سرور مثل رم و CPU هزینه می کنیم که در آخر هم به این نتیجه می رسیم هاست ضعیف هست و مدام هاست گران تر می خریم.این جریان اینقدر برای وب سایت هایی که در حال رشد هستند ، ادامه پیدا می کند تا درجه ای که برخی غیر متخصص بیان می کنند **وردپرس** برای استارت اپ های بزرگ جوابگو نیست

اگر میخواهید با این مشکل رو به رو نشوید به این نمونه دقت کنید:
در بخشی از یک وب سایت فروشگاه کتاب ، لیست سه مورد از آخرین کتاب هایی که به سایت اضافه شده توسط کلاس WP Query نمایش داده می شود مانند عکس زیر

برنامه نویس های وردپرس این کار را به صورت زیر کدنویسی می کنند:

```
$args = array(
        'orderby' => 'ID',
        'order'    => 'DESC',
        'posts_per_page' => 3,
);
$the_query = new WP_Query( $args );
if ( $the_query->have_posts() ) {

        while ( $the_query->have_posts() ) : $the_query->the_post();

        echo '<div>';
        echo '<img alt="'.get_the_post_thumbnail_caption( get_the_ID() ).'"
src="'.get_the_post_thumbnail_url(get_the_ID(),'thumbnail').'">';
        echo'<ahref="'.get_the_permalink().'"
        title="'.get_the_title().'">'.get_the_title().'</a>';
        echo 'نویسنده : '.get_post_meta( get_the_ID(), 'author_book_name',
true );
        echo '<br>';
        echo 'موضوع : ';

        $post_categories = wp_get_post_categories( get_the_ID() );

        foreach ( $post_categories as $category ) {

                echo $categories;

        }

        echo '<div>';
        endwhile;
} else {
        // no posts found
}
wp_reset_postdata();
```

اگر شما هم مثل کد بالا برنامه نویسی وردپرس را انجام دادید میتوانم بگویم که سرعت پایین وردپرس در قالب شما کاملا عادی است.بیایید یک بار پشت صحنه این کد وردپرس را با هم ببینیم. من زمانی که کوئری های دیتابیس وردپرس را دیباگ کردم به نتایج زیر رسیدم.اگر شما هم میخواهید

دستورات SQL وردپرس را در صفحات سایت خود مشاهده کنید مقاله زیر را مطالعه کنید:

حتما بخوانید : اصول اولیه برنامه نویسی افزونه وردپرس برای تولید یک محصول استاندارد

HTTPS://REALWP.NET/WORDPRESS-PLUGIN-DEVELOPMENT-BASICS/

```
SELECT SQL_CALC_FOUND_ROWS wp_posts.ID FROM wp_posts  WHERE 1=1  AND
wp_posts.post_type = 'post' AND (wp_posts.post_status = 'publish')  ORDER
BY wp_posts.ID DESC LIMIT 0, 3
SELECT FOUND_ROWS()
SELECT wp_posts.* FROM wp_posts WHERE ID IN (325,318,311)
SELECT  t.*, tt.*, tr.object_id FROM wp_terms AS t  INNER JOIN
wp_term_taxonomy AS tt ON t.term_id = tt.term_id INNER JOIN
wp_term_relationships AS tr ON tr.term_taxonomy_id = tt.term_taxonomy_id
WHERE tt.taxonomy IN ('category', 'post_tag', 'post_format',
'yst_prominent_words') AND tr.object_id IN (311, 318, 325) ORDER BY t.name
ASC
SELECT post_id, meta_key, meta_value FROM wp_postmeta WHERE post_id IN
(311,318,325) ORDER BY meta_id ASC
SELECT * FROM wp_posts WHERE ID = 327 LIMIT 1
SELECT post_id, meta_key, meta_value FROM wp_postmeta WHERE post_id IN
(327) ORDER BY meta_id ASC
SELECT  t.term_id, tt.parent, tt.count, tt.taxonomy FROM wp_terms AS t
INNER JOIN wp_term_taxonomy AS tt ON t.term_id = tt.term_id INNER JOIN
wp_term_relationships AS tr ON tr.term_taxonomy_id = tt.term_taxonomy_id
WHERE tt.taxonomy IN ('category') AND tr.object_id IN (325) ORDER BY
t.name ASC
SELECT * FROM wp_posts WHERE ID = 324 LIMIT 1
SELECT post_id, meta_key, meta_value FROM wp_postmeta WHERE post_id IN
(324) ORDER BY meta_id ASC
SELECT  t.term_id, tt.parent, tt.count, tt.taxonomy FROM wp_terms AS t
INNER JOIN wp_term_taxonomy AS tt ON t.term_id = tt.term_id INNER JOIN
wp_term_relationships AS tr ON tr.term_taxonomy_id = tt.term_taxonomy_id
WHERE tt.taxonomy IN ('category') AND tr.object_id IN (318) ORDER BY
t.name ASC
SELECT * FROM wp_posts WHERE ID = 313 LIMIT 1
SELECT post_id, meta_key, meta_value FROM wp_postmeta WHERE post_id IN
(313) ORDER BY meta_id ASC
SELECT  t.term_id, tt.parent, tt.count, tt.taxonomy FROM wp_terms AS t
INNER JOIN wp_term_taxonomy AS tt ON t.term_id = tt.term_id INNER JOIN
wp_term_relationships AS tr ON tr.term_taxonomy_id = tt.term_taxonomy_id
WHERE tt.taxonomy IN ('category') AND tr.object_id IN (311) ORDER BY
t.name ASC
```

برای دریافت سه مطلب آخر در وردپرس میانگین ۱۴ کوئری به دیتابیس وردپرس ارسال کرد و این یعنی زنگ خطر کامل.ابتدا با کوئری ، ID سه مطلب آخر را پیدا کرد.و سپس با هر بار اجرا شدن حلقه لوپ در وردپرس به پست ها درخواست زد و نام دسته بندی و پست متاهای آن را دریافت کرد.

بیایید این سناریو را تصور کنید تا راه حل برایمان مشخص شود :

ما در اصل به سه مطلب پایانی وردپرس نیاز داریم که هر کدام دارای ۵ چیز هست :

۱. لینک محتوا
۲. آدرس عکس تصویر شاخص و Caption آن
۳. عنوان مطلب
٤. نام نویسنده کتاب که یک Post meta در دیتابیس هست
۵. نام دسته بندی این مطلب که مرتبط با جدول Taxonomy در دیتابیس هست.

ما میتوانیم این محتویات را در" **نقطه تعیین محتوای وردپرس** "وارد پایگاه داده کنیم و سپس تنها با یک کوئری آن را دریافت کنیم.

من ابتدا یک تابع می نویسم و تنها مواردی که نیاز دارم را از آن بدست می آوریم:

```
function wp_get_the_home_page_content() {
/* Set empty Array For Post*/
$export = array();

$args = array(
       'orderby' => 'ID',
       'order'   => 'DESC',
       'posts_per_page' => 3,
);
$the_query = new WP_Query( $args );
if ( $the_query->have_posts() ) {

       while ( $the_query->have_posts() ) : $the_query->the_post();

       $export[] = array(
              'thumbnail_url' =>
get_the_post_thumbnail_url(get_the_ID(),'thumbnail'),
              'thumbnail_caption' => get_the_post_thumbnail_caption(
get_the_ID() ),
              'post_url' => get_the_permalink(),
              'post_title' => get_the_title(),
              'book_author' => get_post_meta( get_the_ID(),
'author_book_name', true ),
              'category' => wp_get_post_categories( get_the_ID() )
       );

       endwhile;

}
/* Restore original Post Data */
wp_reset_postdata();

/* Set Content to Option Table */
update_option( 'my_theme_home_page', $export, 'no' );

}
```

در این تابع ابتدا کوئری ارسال شد ، سپس مواردی که نیازمند خروجی آن ها در کد Html صفحه بودیم را به آرایه اضافه کردیم.و در آخر در جدول Option با نام my_theme_home_page ذخیره کرده ایم.

اگر میپرسید چرا از Transients API استفاده نکردم کاملا مشخص است چون محتویات ما تاریخ انقضا ندارد.
همانند استاندارد های MVC در برنامه نویسی PHP هیچ گاه سعی نکنید کد Html را نیز به این آرایه اضافه کنید. و بخش View و Model را کاملا جداگانه بررسی و ایجاد کنید.

این تابع برای اینکه با هر بار افزودن مطلب جدید در سایت ما آپدیت شود نیازمند آن هست که در بخشی از وب سایت فراخوانی شود.به نظر شما در چه مرحله ای ما میتوانیم متوجه شویم که اطلاعات وردپرس تغییر کرده است ؟

نقطه تغییر محتوای وردپرس به اکشن و فیلترهایی گفته می شود که در زمان اضافه ، حذف یا ویرایش محتویات پایگاه داده در وردپرس انجام می شود.

در چه زمانی متوجه می شویم یک **پست وردپرس** اضافه یا ویرایش شده است در وردپرس:

```
add_action( 'save_post', 'your_function', 10, 3 );
```

در چه زمانی متوجه می شویم یک پست حذف شده در وردپرس:

```
add_action('before_delete_post','your_function');
add_action('after_delete_post', function($postid) { });
```

در چه زمانی متوجه می شویم یک **دسته وردپرس** ایجاد یا حذف شده :

```
add_action( "delete_category", "your_function");
add_action('create_category', 'your_function', 10, 2);
add_action( "edited_category", "your_function" );
```

در چه زمانی بفهیم روی **برچسب های وردپرس** عملیات ایجاد حذف یا ویرایش انجام شده:

```
add_action( "delete_post_tag", "your_function");
add_action('create_post_tag', 'your_function', 10, 2);
add_action( "edited_post_tag", "your_function" );
```

در چه زمانی متوجه شویم یک **رسانه وردپرس** ایجاد یا حذف شده است:

```
add_action( "add_attachment", "your_function");
add_action("delete_attachment", "your_function");
```

در چه زمانی متوجه شویم یک **فهرست وردپرس** ایجاد یا حذف شده:

```
add_action( "wp_create_nav_menu", "your_function");
add_action("wp_update_nav_menu", "your_function");
```

در چه زمانی یک **دیدگاه وردپرس** ایجاد یا حذف می شود:

```
add_action( 'delete_comment', function($comment_id) { } );
add_action('wp_insert_comment','comment_inserted',99,2);
```

در چه زماتی تنظیمات وردپرس در جدول Option ایجاد یا ویرایش می شوند:

```
add_action('add_option', function( $option_name, $option_value ) {}, 10,
2);
add_action('update_option', function( $option_name, $old_value, $value )
{}, 10, 3);
add_action('deleted_option', function( $option ) {});
```

در چه زمانی یک کاربر وردپرس ایجاد یا حذف می شود:

```
add_action( 'user_register', function($user_id) { }, 10, 1 );
add_action( 'delete_user', function($user_id) { } );
```

در چه زمانی یک ابزارک وردپرس ایجاد یا حذف می شود:

```
add_action( 'wp_register_sidebar_widget', function($widget) { } );
```

و بسیاری از رویداد های دیگر که می توانید در مستندات وردپرس آن ها را براحتی پیدا کنید.این ها همان نقاطی هستند که عملیات تغییر محتوا در بانک اطلاعاتی وردپرس اتفاق می افتد.

اگر به تابع ما دقت کنید متوجه خواهید شد که در چه زمان هایی می بایست تابع فراخوانی شود و عملیات بروز رسانی انجام شود من برای این کار تابع را به رویداد حذف و افزودن مطلب قلاب میکنم به صورت زیر:

```
add_action( 'save_post', 'wp_get_the_home_page_content', 10, 3 );
add_action('before_delete_post','wp_get_the_home_page_content');
```

توسط این کار هر بار که عملیات ویرایش ، افزودن یا حذف مطلب در وردپرس رخ داد محتویات کوئری ما نیز برای نمایش در صفحه سایت بروز رسانی می شود حال تنها نیاز هست برای نمایش در وب سایت از نمونه زیر استفاده کنیم:

```
$post_list = get_option('my_theme_home_page');
foreach($post_list as $e) {
	echo '<div>';
	echo '<img alt="'.$e['thumbnail_caption'].'"
src="'.$e['thumbnail_url'].'">';
	echo '<a href="'.$e['post_url'].'"
title="'.$e['post_title'].'">'.$e['post_title'].'</a>';
	echo 'نویسنده : '.$e['book_author'];
	echo '<br>';
	echo 'موضوع : ';

	foreach ( $e['category'] as $category ) {
		echo $categories;
	}
	echo '<div>';
}
```

این بار دیگر از کوئر های زیاد خبری نیست .تنها با یک کوئری توانستیم اطلاعات مورد نیاز را دریافت کنیم.

همواره در پروژه های بزرگ مبتنی بر وردپرس ، راهکار اصلی کاهش سرعت پایین وردپرس در بازدید های زیاد ، بهینه سازی قالب وردپرس توسط تکنیک های متعدد هست.طراحی قالب تنها زیبایی بصری و گرافیک آن و یا حتی سئو آن نیست. بلکه می بایست در هر خط کدی که می نویسد تمامی عوامل تاثیر گزار را مد نظر خود قرار دهید.