

When to use WP_query(), query_posts() and pre_get_posts

I read @nacin's [You don't know Query](#) yesterday and was sent down a bit of a querying rabbit hole. Before yesterday, I was (wrongly) using `query_posts()` for all my querying needs. Now I'm a little bit wiser about using `WP_Query()`, but still have some gray areas.

What I think I know for sure:

If I'm making *additional* loops anywhere on a page—in the sidebar, in a footer, any kind of "related posts", etc—I want to be using `WP_Query()`. I can use that repeatedly on a single page without any harm. (right?).

What I don't know for sure

1. When do I use @nacin's `pre_get_posts` vs. `WP_Query`? Should I use `pre_get_posts` for everything now?
2. When I want to modify the loop in a template page — lets say I want to modify a taxonomy archive page — do I remove the `if have_posts : while have_posts : the_post` part and write my own `WP_Query`? Or do I modify the output using `pre_get_posts` in my functions.php file?

tl;dr

The tl;dr rules I'd like to draw from this are:

1. Never use `query_posts` anymore
2. When running multiple queries on a single page, use `WP_Query`
3. When modifying a loop, do this _____.

Thanks for any wisdom

Terry

ps: I have seen and read: [When should you use WP_Query vs query_posts\(\) vs get_posts\(\)](#)? Which adds another dimension — `get_posts`. But doesn't deal with `pre_get_posts` at all.

<wp-query> <query-posts> <get-posts> <pre-get-posts>

asked May 1 '12 at 13:04

 saltcod
434 1 5 10

protected by [kaiser](#) Feb 11 at 18:31

This question is protected to prevent "thanks!", "me too!", or spam answers by new users. To answer it, you must have earned at least 10 [reputation](#) on this site.

3 Answers

You are right to say:

Never use `query_posts` anymore

pre_get_posts

`pre_get_posts` is a filter, for altering **any** query. It is most often used to alter only the 'main query':

```
add_action('pre_get_posts','wpse50761_alter_query');
function wpse50761_alter_query($query){

    if( $query->is_main_query() ){
        //Do something to main query
    }
}
```

(I would also check that `is_admin()` returns *false* – though this may be redundant.). The main query appears in your templates as:

```
if( have_posts() ):
    while( have_posts() ): the_post();
    //The Loop
```

```

        endwhile;
    endwhile;
endif;

```

If you ever feel the need to edit this loop – use `pre_get_posts` . i.e. If you are tempted to use `query_posts()` – use `pre_get_posts` instead.

WP_Query

The main query is an important instance of a `WP_Query` object . WordPress uses it to decide which template to use, for example, and any arguments passed into the url (e.g. pagination) are all channelled into that instance of the `WP_Query` object.

For secondary loops (e.g. in side-bars, or 'related posts' lists) you'll want to create your own separate instance of the `WP_Query` object. E.g.

```

$my_secondary_loop = new WP_Query(...);
if( $my_secondary_loop->have_posts() ) :
    while( $my_secondary_loop->have_posts() ) : $my_secondary_loop->the_post();
        //The secondary Loop
    endwhile;
endif;
wp_reset_postdata();

```

Notice `wp_reset_postdata();` – this is because the secondary loop will override the global `$post` variable which identifies the 'current post'. This essentially resets that to the `$post` we are on.

get_posts()

This is essentially a wrapper for a separate instance of a `WP_Query` object. This returns an array of post objects. The methods used in the loop above are no longer available to you. This isn't a 'Loop', simply an array of post object.

```

<ul>
<?php
global $post;
$args = array( 'numberposts' => 5, 'offset'=> 1, 'category' => 1 );
$myposts = get_posts( $args );
foreach( $myposts as $post ) : setup_postdata($post); ?>
    <li><a href="<?php the_permalink(); ?>"><?php the_title(); ?></a></li>
<?php endforeach; wp_reset_postdata(); ?>
</ul>

```

In response to your questions

1. Use `pre_get_posts` to alter your main query. Use a separate `WP_Query` object (method 2) for secondary loops in the template pages.
2. If you want to alter the query of the main loop, use `pre_get_posts`.

edited Nov 7 '12 at 10:30

 **Geert**
3,473 8 18

answered May 1 '12 at 13:27

 **Stephen Harris**
16.1k 3 17 46

2 Thanks for a very good post, Stephen. Really helpful. – [saltcod](#) May 1 '12 at 14:18

2 Excellent, excellent answer. Thank you for being so thorough! – [cale_b](#) May 1 '12 at 14:26

So is there any scenario when one would go straight to `get_posts()` rather than `WP_Query`? – [drtanz](#) Aug 25 '12 at 16:09

@drtanz – yes. Say for instance you don't need pagination, or sticky posts at the top – in these instances `get_posts()` is more efficient. – [Stephen Harris](#) Aug 25 '12 at 18:00

But wouldn't that add an extra query where we could just modify `pre_get_posts` to modify the main query? – [drtanz](#) Aug 26 '12 at 20:54

There are two different contexts for loops:

- **main** loop that happens based on URL request and is processed before templates are loaded
- **secondary** loops that happen in any other way, called from template files or otherwise

Problem with `query_posts()` is that it is secondary loop that tries to be main one and fails miserably. Thus forget it exists.

To modify main loop

- don't use `query_posts()`
- use `pre_get_posts` filter with `$query->is_main_query()` check
- alternately use `request` filter (a little too rough so above is better)



To run secondary loop

Use `new WP_Query` or `get_posts()` which are pretty much interchangeable (latter is thin wrapper for former).

To cleanup

Use `wp_reset_query()` if you used `query_posts()` or messed with global `$wp_query` directly – so you will almost never need to.

Use `wp_reset_postdata()` if you used `the_post()` or `setup_postdata()` or messed with global `$post` and need to restore initial state of post-related things.

edited Nov 7 '12 at 10:32	answered May 1 '12 at 13:27
 Geert 3,473 8 18	 Rarst ♦ 43.4k 3 33 118
<hr/>	
1 Rarst meant <code>wp_reset_postdata()</code> – Gregory Jun 1 '12 at 9:18	
<hr/>	

There is a legitimate scenario for using `query_posts($query)` and that is (for example):

1. You want to display a list of posts or custom-post-type posts on a page (using a page template)
2. You want to make pagination of those posts work

Now why would you want to display it on a page instead of using an archive template?

1. It's more intuitive for an administrator (your customer?) – they can see the page in the 'Pages'
2. It's better for adding it to menus (without the page, they'd have to add the url directly)
3. If you want to display additional content (text, post thumbnail, or any custom meta content) on the template, you can easily get it from the page (and it all makes more sense for the customer too). See if you used an archive template, you'd either need to hardcode the additional content or use for example theme/plugin options (which makes it less intuitive for the customer)

Here's a simplified example code (which would be on your page template – e.g. `page-page-of-posts.php`):

```
/**
 * Template Name: Page of Posts
 */

while(have_posts()) { // original main loop - page content
    the_post();
    the_title(); // title of the page
    the_content(); // content of the page
    // etc...
}

// now we display list of our custom-post-type posts

// first obtain pagination parametres
$paged = 1;
if(get_query_var('paged')) {
    $paged = get_query_var('paged');
} elseif(get_query_var('page')) {
    $paged = get_query_var('page');
}

// query posts and replace the main query (page) with this one (so the pagination
works)
```

```

query_posts(array('post_type' => 'my_post_type', 'post_status' => 'publish', 'paged'
=> $paged))

// pagination
next_posts_link();
previous_posts_link();

// loop
while(have_posts()) {
    the_post();
    the_title(); // your custom-post-type post's title
    the_content(); // // your custom-post-type post's content
}

wp_reset_query() // sets the main query (global $wp_query) to the original page query
(it obtains it from global $wp_the_query variable) and resets the post data

// So, now we can display the page-related content again (if we wish so)
while(have_posts()) { // original main loop - page content
    the_post();
    the_title(); // title of the page
    the_content(); // content of the page
    // etc...
}

```

Now, to be perfectly clear, we could avoid using `query_posts()` here too and use `WP_Query` instead – like so:

```

// ...

global $wp_query;
$wp_query = new WP_Query(array('your query vars here')); // sets the new custom query
as a main query

// your custom-post-type loop here

wp_reset_query();

// ...

```

But, why would we do that when we have such a nice little function available for it?

edited Sep 22 '12 at 12:27



Brian Fegter

6,543 1 12 34

answered Sep 16 '12 at 7:34



Lukas Pecinka

81 1 1

Brian, thanks for that. I've been struggling to get `pre_get_posts` to work on a page in EXACTLY the scenario you describe: client needs to add custom fields/content to what otherwise would be an archive page, so a "page" needs to be created; client needs to see something to add to nav menu, as adding a custom link escapes them; etc. +1 from me! – [Will Lanni](#) Dec 13 '12 at 11:07
