Order posts by custom fields

- Getting started
- Examples
  - get_posts
  - WP_Query
- pre_get_posts
  - functions.php

This article will demonstrate how to retrieve and sort an array of post objects from the database using native WP functions. There are many ways to query posts in WP, however, this article will make use of the common get_posts function, WP_Query Object and pre_get_posts filter.

Getting started

If you are already familiar with the above function, object and filter you may skip this section.

The WP_Query object is used to query posts and will return an object containing an array of $post objects and many useful methods.

The get_posts function makes use of the above WP_Query object, however, it only returns an array of $post objects making it a simpler way to find and loop over posts.

The pre_get_post filter is called after the query object is created, but before the actual query is run.

Examples

get_posts

This example will use the get_posts function to load all the ‘events’ posts ordered by a custom field value of ‘start_date’.

```php
<?php

// get posts
$posts = get_posts(array(
    'post_type'         => 'event',
    'posts_per_page'    => -1,
    'meta_key'          => 'start_date',
    'orderby'           => 'meta_value',
    'order'             => 'DESC'
));

if( $posts ): ?>

    <ul>

    <?php foreach( $posts as $post ):

        setup_postdata( $post )

        ?>
        <li>
            <a href="<?php the_permalink(); ?>"><?php the_title(); ?> (date: <?php
the_field('start_date'); ?>)</a>
        </li>

    <?php endforeach; ?>

    </ul>

    <?php wp_reset_postdata(); ?>

<?php endif; ?>
```

WP_Query

This example will use the WP_Query object to load all the ‘events’ posts ordered by a custom field ‘featured’. In this case, ‘featured’ is a True / False custom field and the result will show featured posts above the rest.

```php
<?php

// query
$the_query = new WP_Query(array(
    'post_type'         => 'event',
    'posts_per_page'    => -1,
    'meta_key'          => 'featured',
    'orderby'           => 'meta_value',
    'order'             => 'DESC'
));

?>
<?php if( $the_query->have_posts() ): ?>
    <ul>
    <?php while( $the_query->have_posts() ) : $the_query->the_post();

        $class = get_field('featured') ? 'class="featured"' : '';

        ?>
        <li <?php echo $class; ?>>
            <a href="<?php the_permalink(); ?>"><?php the_title(); ?></a>
        </li>
    <?php endwhile; ?>
    </ul>
<?php endif; ?>

<?php wp_reset_query();   // Restore global post data stomped by the_post(). ?>
```

pre_get_posts

This example shows how to modify the current posts query. This is useful to modify an existing post type archive. This filter should be used in the functions.php file and requires some logic to avoid modifying all queries.

functions.php

```php
<?php

function my_pre_get_posts( $query ) {
```

```
    // do not modify queries in the admin
    if( is_admin() ) {

        return $query;

    }


    // only modify queries for 'event' post type
    if( isset($query->query_vars['post_type']) && $query->query_vars['post_type'] == 'event' )
{

        $query->set('orderby', 'meta_value');
        $query->set('meta_key', 'start_date');
        $query->set('order', 'DESC');

    }


    // return
    return $query;

}

add_action('pre_get_posts', 'my_pre_get_posts');

?>
```

□□□ #1
□ □□□ □□□ 2026-08-29 15:44:03 UTC
□ □□□ □□□ 2026-08-29 15:44:03 UTC