How to Query Posts by Custom Fields

- Getting Started
  - Example
- Custom Field Parameters
  - Basic Query Example
  - Advanced Query Example
- Examples
  - 1. Single custom field value
  - 2. Multiple custom field text-based values
  - 3. Multiple custom field array-based values
  - 4. Sub custom field values
  - Dynamic $_GET parameters
    - functions.php

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

Getting Started

You can skip this section if you're already familiar with the above function, object, and filter.

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.

Example

This example demonstrates how to query all posts and display them in a list. Please note the functions `setup_postdata()` and `wp_reset_postdata()` are used to allow functions such as `the_permalink()` and `the_title()` to work as expected.

```
<?php

$posts = get_posts(array(
    'posts_per_page'    => -1,
    'post_type'         => 'post'
));

if( $posts ): ?>

    <ul>

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

        setup_postdata( $post );

        ?>
        <li>
            <a href="<?php the_permalink(); ?>"><?php the_title(); ?></a>
        </li>

    <?php endforeach; ?>

    </ul>

    <?php wp_reset_postdata(); ?>

<?php endif; ?>
```

Custom Field Parameters

Both the `get_posts` function and `WP_Query` object accept arguments to perform basic and advanced queries on custom field values. There is both a basic and advanced way to query which are explained below. You can read more about acceptable parameters in the WP codex.

Basic Query Example

This example shows the arguments to find all posts where a custom field called ‘color’ has a value of ‘red’.

```
$posts = get_posts(array(
    'posts_per_page'    => -1,
    'post_type'     => 'post',
    'meta_key'      => 'color',
    'meta_value'    => 'red'
));
```

Advanced Query Example

This example shows arguments to find all posts where a custom field called ‘color’ has a value of ‘red’ or ‘orange’, and another custom field called ‘featured’ (checkbox) is checked.

```
$posts = get_posts(array(
    'posts_per_page'    => -1,
    'post_type'     => 'post',
    'meta_query'    => array(
        'relation'      => 'AND',
        array(
            'key'       => 'color',
            'value'     => array('red', 'orange'),
            'compare'   => 'IN',
        ),
        array(
            'key'       => 'featured',
            'value'     => '1',
            'compare'   => '=',
        ),
    ),
));
```

Examples

Below you will find an assortment of examples. Please note these examples use the `WP_Query` object rather than the `get_posts` function, but the arguments and logic remain the same.

1. Single custom field value

In this example, we will find all posts that have a `post_type` of ‘event’ where the custom field ‘location’ is equal to ‘Melbourne’. The custom field (‘location’ in this case) could be a Text, Radio Button, or Select field (i.e., something that saves a single text value).

```
<?php

// args
$args = array(
    'posts_per_page'    => -1,
    'post_type'     => 'event',
    'meta_key'      => 'location',
    'meta_value'    => 'Melbourne'
);

// query
$the_query = new WP_Query( $args );

?>
<?php if( $the_query->have_posts() ): ?>
    <ul>
    <?php while( $the_query->have_posts() ) : $the_query->the_post(); ?>
        <li>
            <a href="<?php the_permalink(); ?>">
                <img src="<?php the_field('event_thumbnail'); ?>" />
                <?php the_title(); ?>
            </a>
        </li>
    <?php endwhile; ?>
    </ul>
<?php endif; ?>
```

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

2. Multiple custom field text-based values

In this example, we will find all posts that have a `post_type` of ‘event’ where the custom field ‘location’ is equal to ‘Melbourne’, and the custom field ‘attendees’ is higher than 100. The custom field (‘attendees’ in this case) could be a Number, Text, Radio Button, or Select field (i.e., something that saves a single text value).

```
<?php

// args
$args = array(
    'posts_per_page'    => -1,
    'post_type'     => 'event',
    'meta_query'    => array(
        'relation'      => 'AND',
        array(
            'key'       => 'location',
            'value'     => 'Melbourne',
            'compare'   => '='
        ),
        array(
            'key'       => 'attendees',
            'value'     => 100,
            'type'      => 'NUMERIC',
            'compare'   => '>'
        )
    )
);

// query
$the_query = new WP_Query( $args );

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

```
            <a href="<?php the_permalink(); ?>">
                <img src="<?php the_field('event_thumbnail'); ?>" />
                <?php the_title(); ?>
            </a>
        </li>
    <?php endwhile; ?>
    </ul>
<?php endif; ?>

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

3. Multiple custom field array-based values

In this example, we will find all posts that have a `post_type` of ‘event’ where the custom field ‘location’ is equal to ‘Melbourne’ or ‘Sydney’. The custom field (‘location’ in this case) could be a Multi-Select or Checkbox field (i.e., something that saves a serialized array value).

```
<?php

// args
$args = array(
    'posts_per_page'    => -1,
    'post_type'     => 'event',
    'meta_query'    => array(
        'relation'      => 'OR',
        array(
            'key'       => 'location',
            'value'     => 'Melbourne',
            'compare'   => 'LIKE'
        ),
        array(
            'key'       => 'location',
            'value'     => 'Sydney',
            'compare'   => 'LIKE'
        )
    )
);
```

```
// query
$the_query = new WP_Query( $args );

?>
<?php if( $the_query->have_posts() ): ?>
    <ul>
    <?php while ( $the_query->have_posts() ) : $the_query->the_post(); ?>
        <li>
            <a href="<?php the_permalink(); ?>">
                <img src="<?php the_field('event_thumbnail'); ?>" />
                <?php the_title(); ?>
            </a>
        </li>
    <?php endwhile; ?>
    </ul>
<?php endif; ?>

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

4. Sub custom field values

In this example, we will find all events that have a ‘city’ or either ‘Melbourne’ or ‘Sydney’. Each ‘city’ is added as a new row to a repeater field called ‘location’.

To successfully query sub field values, we need to remember that the row number is not known (there may be 1, 2, or even 3 rows of Repeater field data). Therefore, we need to use a LIKE clause in our SQL query to allow for a WILDCARD in the meta_key search. To do this, we create a custom filter to replace the standard ‘=’ with ‘LIKE’.

Update: Since the changed behavior of `esc_sql()` in WordPress 4.8.3, it is not easy to use the `%` character as a placeholder for the following search and replace. Instead, we recommend that you use the `$` character as shown below.

Note: This method requires hooking into the `posts_where` filter, which is not guaranteed to run on all post queries. To overcome this, set `suppress_filters` to false in the argument array passed to `get_posts()` or `WP_Query`.

```
<?php

// filter
```

```
function my_posts_where( $where ) {

    $where = str_replace("meta_key = 'locations_$", "meta_key LIKE 'locations_%", $where);

    return $where;
}

add_filter('posts_where', 'my_posts_where');

// vars
$city = 'Melbourne';

// args
$args = array(
    'posts_per_page'    => -1,
    'post_type'     => 'event',
    'meta_query'    => array(
        'relation'      => 'OR',
        array(
            'key'       => 'locations_$_city',
            'compare'   => '=',
            'value'     => 'Melbourne',
        ),
        array(
            'key'       => 'locations_$_city',
            'compare'   => '=',
            'value'     => 'Sydney',
        )
    )
);

// query
$the_query = new WP_Query( $args );

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

Dynamic $_GET parameters

This example shows how to use `$_GET` parameters (from the URL) to modify the query of a post type archive. This example assumes a post type exists for ‘event’ and that its archive exists at the url `www.website.com/events`.

The event post type contains a select field called ‘city’ with values such as ‘melbourne’ and ‘sydney’. By adding a parameter to the url, the query will be modified and only posts that match the ‘city’ will be shown: `www.website.com/events?city=melbourne`.

functions.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' )
{

        // allow the url to alter the query
        if( isset($_GET['city']) ) {

            $query->set('meta_key', 'city');
            $query->set('meta_value', $_GET['city']);

        }
```

```
    }



    // return
    return $query;

}

add_action('pre_get_posts', 'my_pre_get_posts');
```

□□□ #2
□ □□□ □□□ 2026-08-29 15:44:03 UTC
□ □□□ □□□ 2026-08-29 15:44:03 UTC