FUN WITH WP_QUERY

Erick Hitter
@ethitter

Lead WordPress Developer at Oomph, Inc.
WordPress 3.3 Core Contributor
WordCamp Boston Organizer
Plugin Author

WHAT IS WP_QUERY?

- WordPress class defined in wp-includes/query.php.
- Responsible for majority of post-type retrieval from database.
- Any front-end request made to WordPress is fulfilled by WP_Query in what I refer to as the *main query*.
- Used extensively by WordPress but also available to developers.

USING WP_QUERY

Most often, this is done in "The Loop" to use the results from the main query.

```
<?php
        if ( have_posts() ) {
                while( have_posts() ) {
                        the_post();
                }
        }
?>
```

USING WP_QUERY: `NEW WP_QUERY();`

```
<?php
        $queried_posts = new WP_Query();

        if ( $queried_posts->have_posts() ) {
                while( $queried_posts->have_posts() ) {
                        $queried_posts->the_post();
                        //Use functions such as the_title() and the_content() here.
                }
        }

        wp_reset_query();
        wp_reset_postdata();
?>
```

- Creates a new object of the type WP_Query, setting up another instance of The Loop.
- `wp_reset_query()` and `wp_reset_postdata()` are necessary to restore the main WordPress query. More on this later.
- Best used whenever Template Tags are needed.

USING WP_QUERY: `GET_POSTS();`

```
<?php
        $queried_posts = get_posts();

        foreach( $queried_posts as $queried_post ) {
                setup_postdata( $queried_post );
        }

        wp_reset_postdata();
?>
```

- `get_posts()` generates an array of post objects that match the query parameters.
- `setup_postdata()` and `wp_reset_postdata()` can be used if Template Tags, such as `the_title()`, are needed.
- Best used when Template Tags aren't needed or post data will be repurposed, such as generating an array of post IDs.

RESETTING QUERY AND POSTDATA

- After using `new WP_Query()`, calling `wp_reset_query()` and `wp_reset_postdata()` restores the original query and postdata.
- After using `setup_postdata()`, calling `wp_reset_postdata()` restores the post.
- Calling the two reset functions ensures that the Conditional Tags operate as expected and balance of page renders properly.

QUERY BASICS

- Can pass parameters as an array or query string.

```
<?php
        $foo = new WP_Query( array( 'posts_per_page' => 5 ) );
        $bar = new WP_Query( 'posts_per_page=5' );
?>
```

- Array format is more flexible and necessary for queries such as:
    - 'post__not_in'
    - 'tax_query'
    - 'meta_query'

- Query string format is parsed into an array by the `WP_Query` class.

QUERY PARAMETERS

- Extensive list in the Codex at http://codex.wordpress.org/Class_Reference/WP_Query#Parameters
- Knowing default parameters for query reduces duplication.
  - 'post_type' => 'post'
  - 'orderby' => 'date'
  - 'order' => 'DESC'
  - 'posts_per_page' => *Value set under Settings -> Reading*
- Can query for (among many others):
  - Author
  - Date/time
  - Post type
  - Taxonomy term assignments
  - Meta data

SAMPLE QUERY 1

Five most recent posts

```
<?php $foo = new WP_Query( array(
        'posts_per_page' => 5
) ); ?>
```

SAMPLE QUERY 2

Five most recent posts by author with ID *15*

```
<?php $foo = new WP_Query( array(
        'posts_per_page' => 5,
        'author' => 15
) ); ?>
```

SAMPLE QUERY 3

Five most recent posts by author with ID *15* in the *toast*

```
<?php $foo = new WP_Query( array(
        'posts_per_page' => 5,
        'author' => 15,
        'category_name' => 'toast'
) ); ?>
```

SAMPLE QUERY 4

Five most recent posts or pages

```
<?php $foo = new WP_Query( array(
        'posts_per_page' => 5,
        'post_type' => array( 'post', 'page' )
) ); ?>
```

USING THE `TAX_QUERY`

- Introduced in WordPress 3.1
- Permits multiple taxonomy queries within a single `WP_Query`.
- **Important to remember that the `tax_query` parameter takes an array of arrays, even if querying for a single taxonomy term.**
- Each query accepts up to five arguments:
  - taxonomy (required) - string
  - field - `term_id` or `slug`
  - terms (required) - string or array
  - include_children - boolean, defaults to `true`
  - operator - string, either `AND`, `IN`, or `NOT IN`

USING THE `TAX_QUERY`

Using the *toast* example from Sample Query 3:

```
<?php $foo = new WP_Query( array(
        'posts_per_page' => 5,
        'author' => 15,
        'category_name' => 'toast'
) ); ?>
```

Becomes

```
<?php $foo = new WP_Query( array(
        'posts_per_page' => 5,
        'author' => 15,
        'tax_query' => array( array(
                'taxonomy' => 'category',
                'field' => 'slug',
                'terms' => 'toast'
        ) )
) ); ?>
```

USING THE `TAX_QUERY`: MULTIPLE QUERIES

Querying two categories:

```
<?php $foo = new WP_Query( array(
        'posts_per_page' => 5,
        'author' => 15,
        'tax_query' => array( array(
                'taxonomy' => 'category',
                'field' => 'slug',
                'terms' => array( 'toast', 'breakfast' ),
                'operator' => 'AND'
        ) )
) ); ?>
```

Querying a category and a tag:

```
<?php $foo = new WP_Query( array(
        'posts_per_page' => 5,
        'author' => 15,
        'tax_query' => array(
                'relation' => 'AND',
                array(
                        'taxonomy' => 'category',
                        'field' => 'slug',
                        'terms' => 'toast'
                ),
                array(
                        'taxonomy' => 'post_tag',
                            'field' => 'slug',
                            'terms' => 'wheat'
                    )
            )
) ); ?>
```

USING THE `TAX_QUERY`: MULTIPLE QUERIES

Querying a category and a tag, excluding subcategories:

```
<?php $foo = new WP_Query( array(
        'posts_per_page' => 5,
        'author' => 15,
        'tax_query' => array(
                'relation' => 'AND',
                array(
                        'taxonomy' => 'category',
                        'field' => 'slug',
                        'terms' => 'toast',
                        'include_children' => false
                ),
                array(
                        'taxonomy' => 'post_tag',
                        'field' => 'slug',
                        'terms' => 'wheat'
                )
        )
) ); ?>
```

USING THE `META_QUERY`

- Introduced in WordPress 3.1, along with `tax_query`
- Permits multiple post meta queries within a single `WP_Query`.
- **Like the `tax_query`, it is important to note that the `meta_query` parameter takes an array of arrays.**
- Each query accepts up to four arguments:
  - key (required) - string
  - value - string or array
  - compare - string, such as `=`, `!=`, or `IN`. Full list on the WP_Query Codex page.
  - type - string, such as `NUMERIC` or `DATE`. Full list on the WP_Query Codex page.

USING THE `META_QUERY`

- Five posts with Featured Images

```
<?php $foo = new WP_Query( array(
        'posts_per_page' => 5,
        'meta_query' => array( array(
                'key' => '_thumbnail_id'
        ) )
) ); ?>
```

- Five posts using the same thumbnail, ID 100

```
<?php $foo = new WP_Query( array(
        'posts_per_page' => 5,
        'meta_query' => array( array(
                'key' => '_thumbnail_id',
                'value' => 100,
                'type' => 'NUMERIC'
        ) )
) ); ?>
```

USING THE `META_QUERY`: MULTIPLE QUERIES

- Five posts with Featured Images **AND** flagged as "featured" posts

```
<?php $foo = new WP_Query( array(
        'posts_per_page' => 5,
        'meta_query' => array(
                array(
                        'key' => '_thumbnail_id'
                ),
                array(
                        'key' => '_featured'
                )
        )
) ); ?>
```

- Five posts with Featured Images **OR** flagged as "featured" posts

```
<?php $foo = new WP_Query( array(
        'posts_per_page' => 5,
        'meta_query' => array(
                'relation' => 'OR',
                array(
                        'key' => '_thumbnail_id'
                ),
                array(
                        'key' => '_featured'
                )
        )
) ); ?>
```

GET_QUERY_VAR();

- Used to retrieve a query parameter from the main query.

-
  ```
  <?php
          $qty = get_query_var( 'posts_per_page' );
          $post_type = get_query_var( 'post_type' );
  ?>
  ```

IS_MAIN_QUERY();

- Introduced in WordPress 3.3
- Simple conditional tag that determines whether or not the current query is the main WordPress query.

~~QUERY_POSTS();~~

- Evil WordPress function provided to modify or override main query.
- Accepts query arguments as described in the preceeding slides.
- Used within a theme file (archive.php, home.php, index.php, single.php, etc).
- When called, original main query is replaced by the query specified by `query_posts()`.
- Introduces unnecessary performance degredation.

Thankfully, a far-superior alternative exists!

THE *PRE_GET_POSTS* ACTION

- Action is executed right before `WP_Query` parses the query arguments.
- Allows any and all query arguments to be modified before a database query is ever run.
- Can be used to universally modify queries on a WordPress site, or to modify specific queries.
- Rather than placing modifications in individual theme files as is done with `query_posts()`, changes can be centralized in functions.php.

USING THE *PRE_GET_POSTS* ACTION

```
<?php
        function bwpm_pre_get_posts( $query ) {
                //Herein, modify the query
        }

        add_action( 'pre_get_posts', 'bwpm_pre_get_posts' );
?>
```

The `$query` variable contains an object with two helpful methods:

- `get( $query_parameter )` - retrieves a specified query parameter from the current query.
- `set( $query_parameter, $value )` - sets a specified query parameter in the current query.

USING THE *PRE_GET_POSTS* ACTION

Show only five posts on any and every archive page:

```
<?php
        function bwpm_pre_get_posts( $query ) {
                $query->set( 'posts_per_page', 5 );
        }

        add_action( 'pre_get_posts', 'bwpm_pre_get_posts' );
?>
```

To do the same on just the front page:

```
<?php
        function bwpm_pre_get_posts( $query ) {
                if ( $query->is_home() )
                        $query->set( 'posts_per_page', 5 );
        }

        add_action( 'pre_get_posts', 'bwpm_pre_get_posts' );
?>
```

USING THE *PRE_GET_POSTS* ACTION

To do the same only when a `posts_per_page` value isn't set (is using the WordPress default):

```
<?php
        function bwpm_pre_get_posts( $query ) {
                if ( ! $query->get( 'posts_page_page' ) )
                        $query->set( 'posts_per_page', 5 );
        }

        add_action( 'pre_get_posts', 'bwpm_pre_get_posts' );
?>
```

QUESTIONS?