WP Query with Elementor

PHP, Wordpress

Web Skills : 3

When it is time to sort or simply display posts according to certain criteria, you need to use queries. With Wordpress, there is an integrated tool called WP Query. You just have to create your query and Wordpress takes care of the rest.

Published Friday March 27th 2020, 15:07
Modified Monday August 26th 2024, 10:05

By Olivier Paudex

Introduction

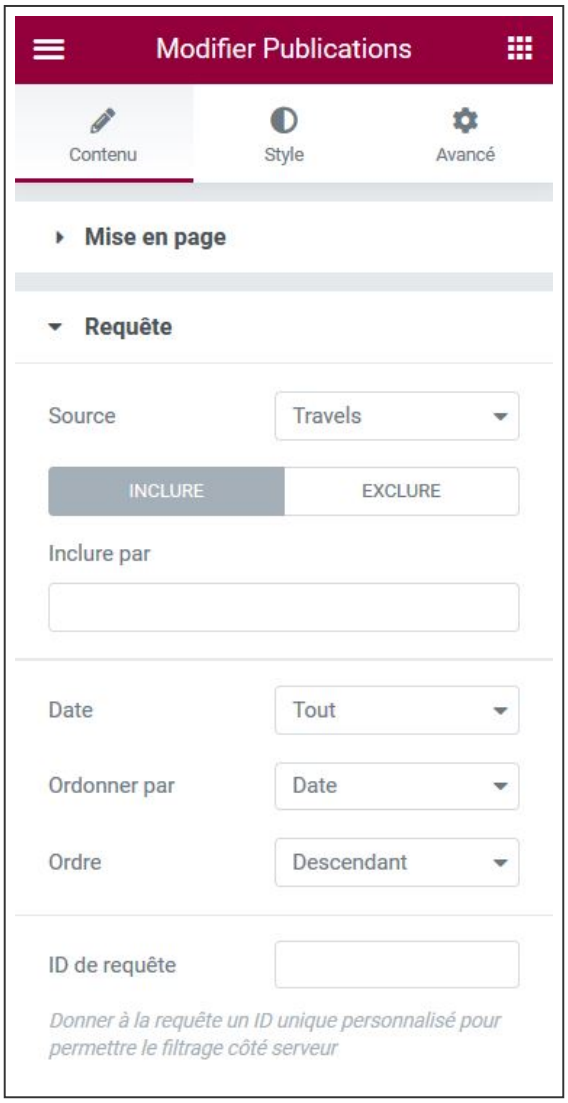
L'outil WP Query est intégré à WordPress et se profile comme le meilleur moyen d'obtenir un résultat rapide quand vient le moment d'afficher, de trier, ou de répondre à certains critères de recherche. Et Elementor peut les utiliser très simplement avec son widget **"Publications"**.

Afficher les publications

Pour afficher les publications sur une page, il suffit de glisser le widget publications dans une section

Le widget publications

Le but de ce chapitre n'étant pas de mettre en page les publications, on passe directement à l'onglet " **Requête**". Dans l'exemple ci-dessous, le site va afficher toutes les publications qui appartiennent au type personnalisé "**Voyages**", ou "**Travels**" en anglais.



L'onglet requête du widget publication

Les critères de filtrage restent très simple. On peut jouer sur la taxonomie et l'ordre croissant ou décroissant des publications, et c'est à peu près tout. Si on désire afficher certaines publications seulement, par exemple, celles dont le titre contient un certain mot, on ne peut pas le faire et c'est là que l'outil WP Query entre en ligne de compte.

Créer une requête avec WP Query

Pour créer une requête avec WP Query, il faut utiliser son thème enfant et créer un nouveau fichier nommé ici "**Custom_Filters.php**"

Le nouveau fichier vient s'ajouter dans le fichier "**functions.php**".

```
// Personnal include
$dir = get_stylesheet_directory();
// Posts filtersinclude_once ($dir . '/php/Custom_Filters.php');
```

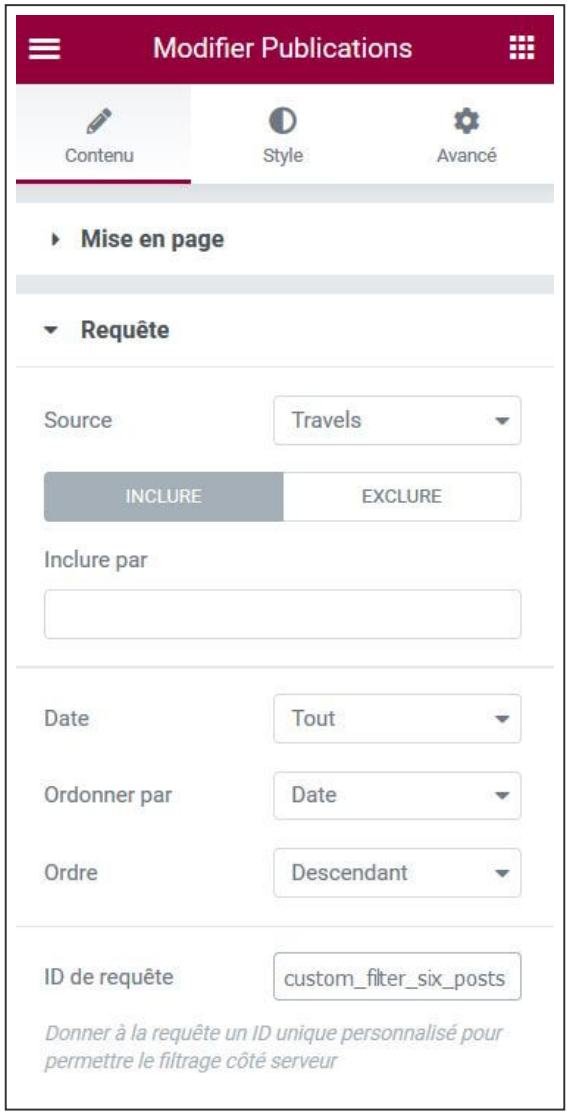
Dans le fichier **"Custom_filter.php"**, on écrit une requête en utilisant une action Elementor : **elementor/query/nom_de_requête**. Par exemple, ci-dessous, la requête va afficher les 6 dernières publications du type personnalisés **"Voyages"**, qui sont dans l'état **"publié"**.

La fonction se nomme **"last_six_posts"** et la requête se nomme **"custom_filter_six_posts"**. C'est cette dernière qu'il faudra utiliser pour lier la requête au widget Elementor.

```
// Showing 6 last posts written
function last_six_posts ($query) {
  $query->set('post_status', 'publish');
  $query->set('post_type', ['travels']);
  $query->set('posts_per_page', 6);
  $query->set('order', 'desc');
}add_action('elementor/query/custom_filter_six_posts', 'last_six_posts');
```

Lier une requête avec Elementor

Une fois la requête écrite, il faut tout simplement la lier en introduisant son nom dans **ID de requête** du widget publication, comme dans l'image ci-dessous.



l'ID de requête du widget publication

Quelques exemples un peu plus complexe

Voici un exemple qui utilise les "**meta query**", qui ne sont rien d'autres que des champs créés avec le plugin ACF. La requête ci-dessous va afficher les publications dont le pays est "**argentine**" ou "**egypte**".

```
// Showing posts filtered by meta query
function meta_posts ($query) {
  $meta_query[] = [
    'key' => 'country',
    'value' => ['argentine','egypte'],
    'compare' => 'in',
  ];
  $query->set('post_status', 'publish');
  $query->set('post_type', ['travels']);
```

```
  $query->set('meta_query', $meta_query);
  $query->set('order', 'desc');
}add_action( 'elementor/query/custom_filter_meta_posts', 'meta_posts' );
```

Un autre exemple, très complexe celui-ci, dont le but est d'afficher les autres publications correspondantes à celle qui est présentement à l'écran, dont le type et les termes sont identiques. On appelle celles-ci les **publications connexes** ou en anglais, **"related posts"**. En plus, la requête va afficher les dix dernières publications dans un ordre aléatoire.

```
// Showing all related posts, regarding the type and terms
function related_posts ($query) {
  global $post;

  // Get custom post type from current post
  $type = get_post_type($post->ID);
  // Get current post
  $post = get_post($post->ID);
  // Get all taxonomies of the custom post
  $taxonomies = get_object_taxonomies($post);
  foreach ($taxonomies as $taxonomy) {
    // Get all terms from current post
    $terms = get_the_terms($post->ID, $taxonomy);
    // If no terms, continue to next one
    if (empty($terms)) continue;
    // Get all slug terms
    $term_list = wp_list_pluck( $terms, 'slug' );
    // Create the taxonomy query. Must be an array of array
    $tax_query = array(
      array(
        'taxonomy' => $taxonomy,
        'field'   => 'slug',
        'terms'    => $term_list
      )
    );
  }
  // Get the related posts
  $query->set('post_status', 'publish');
  $query->set('post_type', $type);
  $query->set('posts_per_page', 10);
  $query->set('post__not_in', [$post->ID]);
  $query->set('orderby','rand');
  $query->set('tax_query', $tax_query);
}add_action( 'elementor/query/custom_filter_related_posts', 'related_posts' );
```