WordPress : requêtes personnalisées WP Query

WP Query : définition

WP Query est une classe de WordPress qui permet d'effectuer des **requêtes personnalisées sur la base de données** pour récupérer des contenus spécifiques selon certains critères.

Elle est utilisée pour récupérer des articles, des pages, des types de contenu personnalisés, des commentaires, des utilisateurs, etc.

WP Query offre une grande flexibilité pour personnaliser les résultats de la requête en fonction des besoins spécifiques du site.

WP Query fonctionne sans que vous ayez à écrire de **requête SQL**.

WP Query : utilisation

Exemple d'usage : vous souhaitez afficher les quatre derniers articles de la catégorie "actualités" sur la page d'accueil de votre site web.

Remarque : vous pourriez tout aussi bien vouloir retourner des **CPT (Custom Post Type)**, comme des recettes de cuisine, des formations….

Voici comment fonctionne WP Query et comment vous pouvez créer des requêtes personnalisées dans WordPress :

1. **Instanciation de WP Query :** pour créer une nouvelle requête, vous devez instancier la classe **WP Query** en utilisant le constructeur. Vous pouvez définir différents paramètres pour personnaliser votre requête, tels que les types de contenu à récupérer, les critères de recherche, les relations entre les critères, le nombre de résultats à retourner, etc.
2. **Définir les arguments de la requête :** vous devez spécifier les arguments de la requête en utilisant les différents paramètres disponibles.
3. **Exécution de la requête :** une fois que vous avez défini les arguments de la requête, vous devez exécuter la requête. Cela récupérera les résultats de la requête sous la forme d'un tableau d'objets de type `WP_Post`.
4. **Boucle sur les résultats :** après avoir obtenu les résultats, vous pouvez les parcourir à l'aide d'une boucle pour afficher les contenus correspondants dans votre thème WordPress.

WP Query : arguments

Les arguments les plus couramment utilisés incluent :

- **'post_type' :** pour spécifier le type de contenu à récupérer (article = **post**, **page**, type de contenu personnalisé, etc.).

- **‘post_status’ :** pour spécifier l’état des contenus à récupérer (publié, brouillon, en attente de relecture, etc.).
- **‘posts_per_page’ :** nombre d’éléments à récupérer.
- **‘category_name’ :** le **slug** de la catégorie, pour récupérer des contenus basés sur le nom d’une catégorie.
- **‘cat’ :** l’ID de la catégorie, pour récupérer des contenus basés sur l’identifiant d’une catégorie.
- **‘tag’ :** pour récupérer des contenus basés sur un ou plusieurs mots-clés (étiquettes).
- **‘author’ :** pour récupérer des contenus créés par un auteur spécifique.
- **‘s’ :** pour lancer une recherche sur un mot-clé.
- **‘meta_key’ et ‘meta_value’ :** pour récupérer des contenus basés sur des champs personnalisés et leur valeur associée.
- **‘orderby’ et ‘order’ :** pour trier les résultats en fonction de certains critères (date, titre, auteur, etc.) avec **‘orderby’** ayant pour valeur **ASC** (ascendant) ou **DESC** (descendant).

Info : il existe plus d’une cinquantaine d’arguments pouvant être utilisés comme paramètres de WP Query. Il vous faudra voir directement dans la documentation officielle.

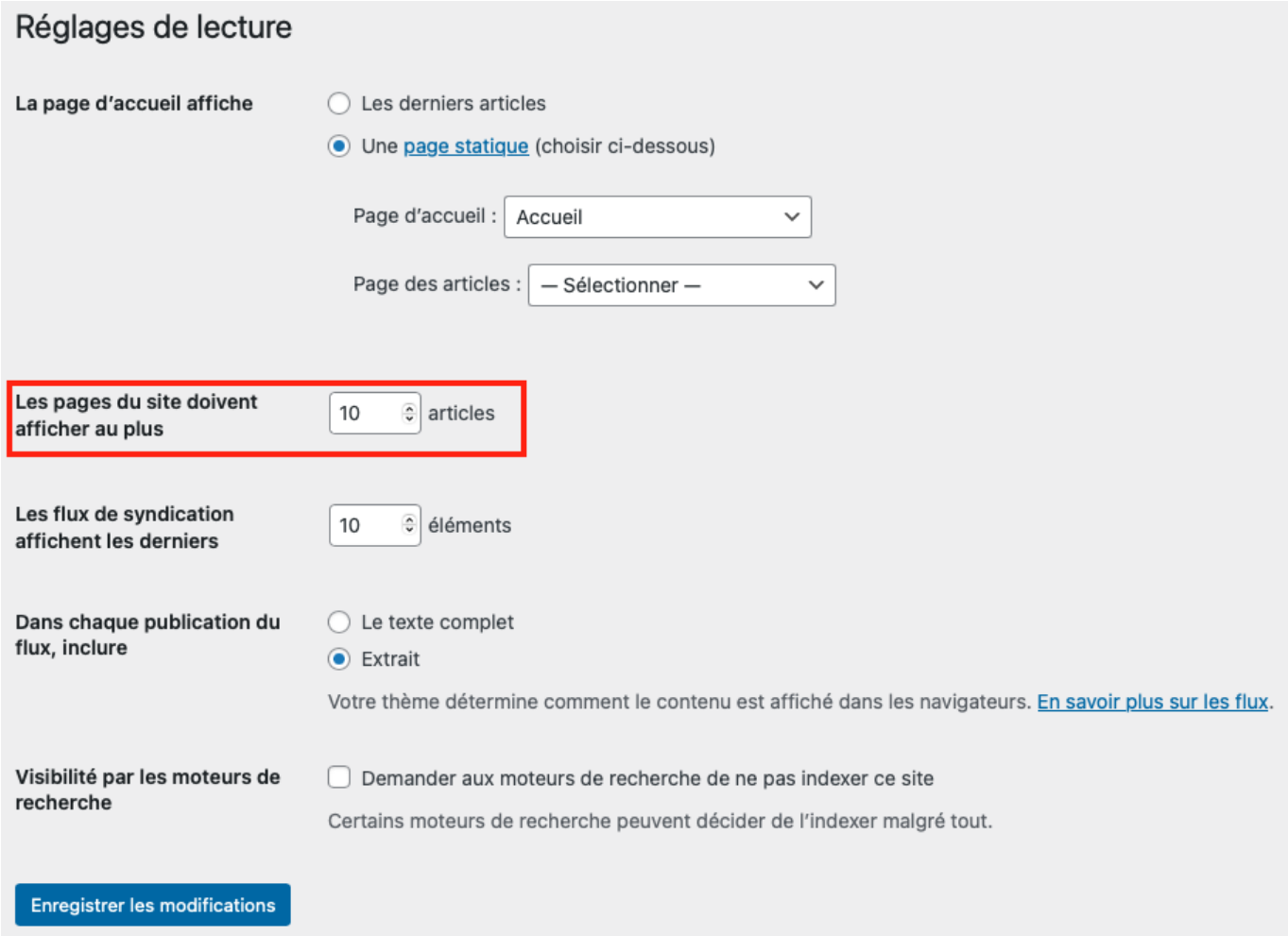
WP Query : exemple complet

```
// on définit les argument pour définir les enregistements à récupérer
$args = array(
    'post_type' => 'post',
    'post_status' => 'publish',
    'category_name' => 'actualites',
    'posts_per_page' => 5,
    'orderby' => 'date',
    'order' => 'DESC'
);

// on exécute la requête WP Query
$query = new WP_Query($args);

// boucle WordPress
if ($query->have_posts()) {
    while ($query->have_posts()) {
        $query->the_post();
        // afficher le contenu de l'article ici (par exemple, le titre et le
contenu).
        // the_title();
        // the_content();
        // the_post_thumbnail();
    }
    // on réinitialise la requête principale (important)
    wp_reset_postdata();
} else {
    // aucun article trouvé.
}
```

Remarque : la boucle utilise les mêmes **templates tags** que la boucle principale.

Nombre d’éléments retournés et pagination

Remarque : par défaut, la requête renvoie le nombre d’enregistrements définis dans **Réglages > Lecture** (initialement configuré à 10).



Vous pouvez modifier cette valeur.

Exemple :

```
<?php
$args = array(
    'posts_per_page' => get_option( 'posts_per_page'), // valeur par défaut
    // ...
);
$query = new WP_Query( $args );
```

```
<?php
$args = array(
    'posts_per_page' => 5, // 5 résultats
    // ...
);
$query = new WP_Query( $args );
```

```
<?php
$args = array(
    'posts_per_page' => -1, // tous les résultats
```

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

Vous pouvez également ajouter le paramètre **'offset'** afin de gérer la **pagination** (ex. 10 articles par pages, mais en commençant à la page 3 = les articles 21 à 30).

```
<?php

$args = array(
    'posts_per_page' => 10, // 10 enregistrement retournés
    'offset' => 20, // à partir du vingtième
);

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

Remarque : WordPress génère lui-même la **requête SQL**.

Requête personnalisée et boucle principale

Exemple :

```
<?php

get_header();

// boucle principale
if( have_posts() ) : while( have_posts() ) : the_post();
        the_title(); // titre de la page (boucle principale)
        $my_query = new WP_Query( array( 'post_type' => 'post' ) );
        // requête personnalisée
        if( $my_query->have_posts() ) : while( $my_query->have_posts() ) :
                $my_query->the_post();
                the_title(); // titre de l'article courant (requête
personnalisée)
        the_content(); // contenu de l'article courrant (requête
personnalisée)

        endwhile; endif;
        wp_reset_postdata(); // réinitialisation des données

        the_content();

endwhile; endif;

get_footer();
```

Remarque : l'utilisation de **`wp_reset_postdata()`** après la boucle est importante pour réinitialiser les données de la requête personnalisée afin de ne pas interférer avec les autres requêtes sur la page et revenir à la boucle principale.

Gestion des dates

Pour utiliser WP Query avec des filtres par dates, vous pouvez utiliser les paramètres **'date_query**' pour spécifier les conditions de date que vous souhaitez appliquer à votre requête personnalisée. Ces filtres vous permettront de récupérer des contenus publiés à des dates spécifiques, compris entre certaines dates, avant ou après une date donnée, etc.

Récupérer les contenus publiés après une date spécifique

```
$date_query = array(
    array(
        'after'     => '2023-01-01', // remplacez cette date par la date
souhaitée
        'inclusive' => true,
    ),
);

$args = array(
    'post_type'   => 'post', // remplacez par le type de contenu souhaité
    'post_status' => 'publish',
    'date_query'  => $date_query,
    // autres paramètres de requête (si nécessaire)
);

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

Récupérer les contenus publiés avant une date spécifique

```
$date_query = array(
    array(
        'before'    => '2023-07-01', // Remplacez cette date par la date
souhaitée
        'inclusive' => true,
    ),
);

$args = array(
    'post_type'   => 'post', // remplacez par le type de contenu souhaité
    'post_status' => 'publish',
    'date_query'  => $date_query,
    // autres paramètres de requête (si nécessaire)
);

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

Récupérer les contenus publiés entre deux dates spécifiques

```
$date_query = array(
    array(
        'after'     => '2023-01-01', // Remplacez cette date par la date de
début souhaitée
        'before'    => '2023-07-01', // Remplacez cette date par la date de
```

```
fin souhaitée
        'inclusive' => true,
    ),
);

$args = array(
    'post_type'   => 'post', // remplacez par le type de contenu souhaité
    'post_status' => 'publish',
    'date_query'  => $date_query,
    // autres paramètres de requête (si nécessaire)
);

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

Connaitre le nombre d’articles retournés par la WP Query

Pour connaître le nombre d’articles retournés par la WP Query, vous pouvez utiliser la méthode **$query->found_posts**.

Cette propriété renvoie le nombre total d’articles qui correspondent aux critères de votre requête personnalisée.

```
$args = array(
    'post_type'      => 'post', // remplacez par le type de contenu souhaité
    'post_status'    => 'publish',
    'posts_per_page' => 10,
    // autres arguments de requête (le cas échéant)
);

$query = new WP_Query($args);

if ($query->have_posts()) {
    // boucle des résultats
    while ($query->have_posts()) {
        $query->the_post();
        // afficher le contenu de l'article ici (par exemple, le titre et le
contenu).
    }
    wp_reset_postdata();

    // nombre total d'articles retournés par la requête
    $total_posts = $query->found_posts;
    echo 'Nombre total d'articles : ' . $total_posts;
} else {
    echo 'Aucun article trouvé.';
}
```