Formulaire et filtre de recherche

Elementor, SQL

Web Compétences : 5

Une page d'archives sans filtre de recherche n'est pas vraiment une page d'archives. Dans tous les cas, ce n'est pas bien pratique. C'est pourquoi, ce chapitre vous guide dans la réalisation d'un formulaire Elementor qui débouche sur un moteur de recherche créé en SQL. Attention, haut niveau !

Publié lundi 18 novembre 2019, 13h29
Modifié lundi 26 août 2024, 10h06

By Olivier Paudex

Introduction

Comme point de départ à la réalisation d'un filtre de recherche, il n'y a pas d'autres options que d'élaborer un formulaire où le visiteur peut entrer des critères selon sa requête. Et dans cette voie, Elementor nous facilite la tâche en fournissant dans sa boîte d'outil, le widget '**formulaire**". Une fois cette tâche relativement facile ajoutée à une page d'archives, le vrai défi peut commencer. Il va falloir créer une requête qui va filtrer et retourner le résultat attendu par le visiteur. Deux choix se proposent à vous. Le premier est le moteur de requêtes embarquées de WordPress, baptisé '**WP Query**". Le deuxième est beaucoup plus classique et permet aussi d'aller beaucoup plus loin, c'est le **SQL (Structured query language)**. Ce site web utilise le SQL, langage qui n'a pratiquement aucune limite quant il faut questionner une base de données.

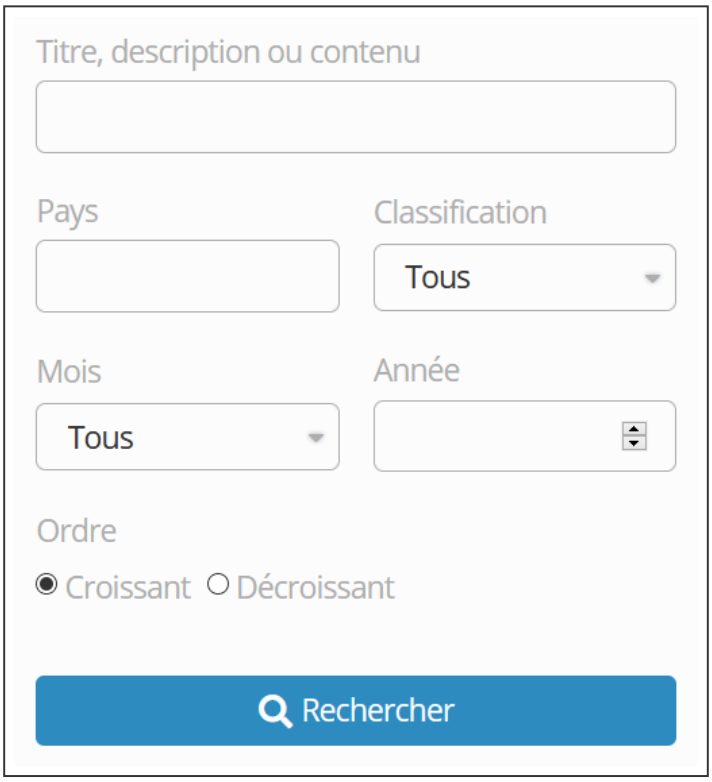
Comment créer un formulaire ?

Le widget **"formulaire"** appartient à la version **Elementor Pro**. Tout mon blog utilise les fonctionnalités

de la version pro. Il existe d'autres plugins pour créer des formulaires, mais un des avantages d'Elementor, c'est qu'il est déjà présent dans le constructeur de pages et que celui-ci peut exécuter du code PHP à sa fermeture.

Le widget formulaire

Créer un formulaire avec **Elementor**, c'est aussi simple que de glisser-déposer le widget dans une section, d'ajouter les champs selon le but à atteindre et de configurer sa fonction. Ci-dessous, le formulaire de ma section voyages.



Filtre de recherche de la section voyages

Définir les champs

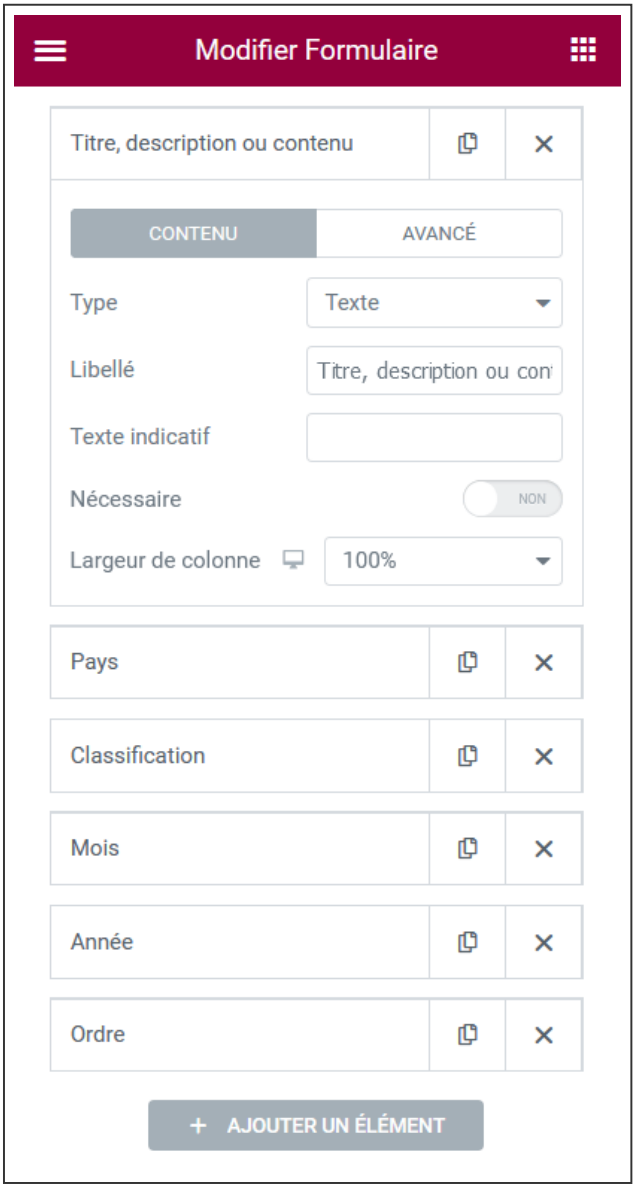
Quand on parle de formulaire, qu'il soit créé à l'aide d'Elementor ou en pure HTML, la première mission va être de définir les champs. En effet, le visiteur qui va vouloir lancer une recherche dans les publications du site le fera dans une certaine logique. Mais personne ne pense comme tout le monde. Définir les champs et créer un formulaire adéquat peut demander beaucoup de temps et de réflexions.

> Sur le point de la définition des champs, quand on le fait pour un client, l'auteur du site web va se lancer dans une certaine logique et se mettre à la place du visiteur pour se poser les vrais questions. Dans la réalisation d'un projet, une discussion va souvent avoir lieu avec le client et c'est le rôle de l'analyste, de poser les bonnes questions afin de savoir sur quoi on veut filtrer les données et ainsi définir les bons champs. On appelle cela "**élicitation de personnes**". Cette tâche se déroule pendant la conception et la réalisation du projet. On l'appelle souvent "**conception fonctionnelle**".

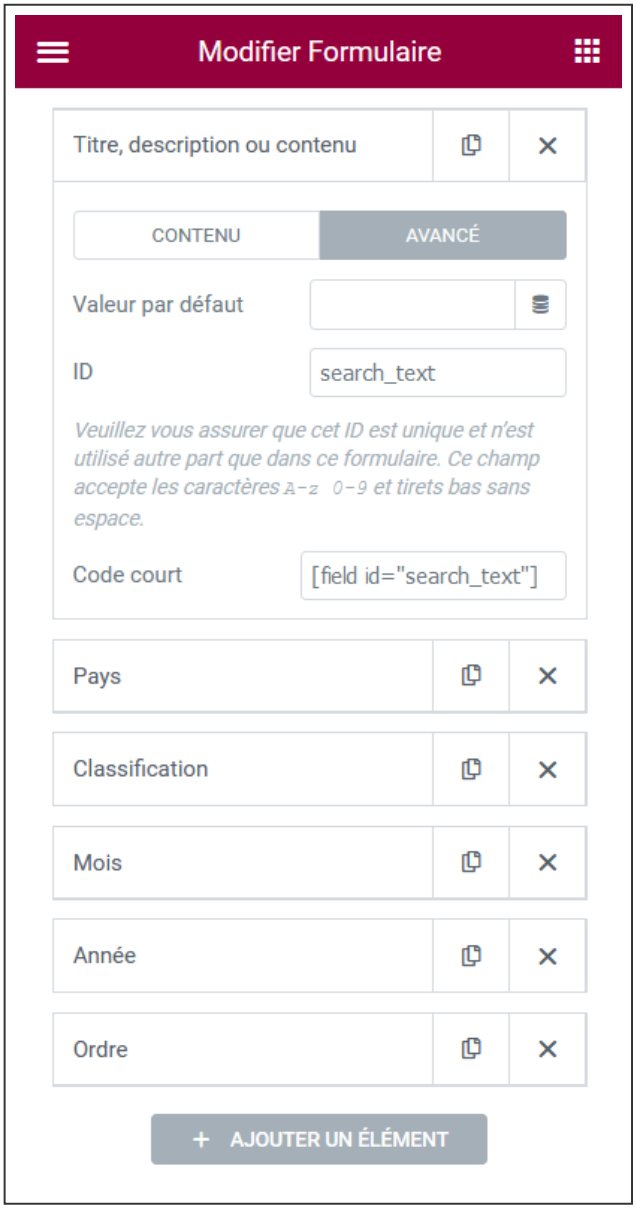
Dans ce site web, le formulaire de la rubrique de voyages est relativement simple. Elle contient :

- Un champ de recherche sur le titre, le contenu ou la description (l'extrait). Il est de type "**chaîne de caractères**" ou "**string**" en anglais.
- Un champ de recherche sur le pays où s'est déroulé le voyage. Lui aussi est un "**string**".
- Un champ de recherche sur la classification (ou taxonomie) du voyage. Encore un "**string**".
- Deux champs de recherche permettant de filtrer le mois et l'année. Le mois est un "**string**" et l'année est de type "**numérique entier**".
- Une option permettant de classer les publications dans l'ordre croissant ou décroissant. Aussi de type "**string**".

- Créez tous les champs (ci-dessous, l'exemple de mon type **"voyages"**).
- Dans l'onglet "**contenu**", paramétrez le type, le libellé du champ et d'autres notions si nécessaire.
- Dans l'onglet "**avancé**", donnez un **ID** à votre champ (ci-dessous, le nom du champ description sera "**search_text**").



L'onglet contenu



L'onglet avancé

Lier le code et le formulaire

Pour lier le formulaire précédemment ajouté, il faut lui donner un nom. Pour celui-ci, son nom est " **filter_travels**". A ne pas confondre avec le nom de la fonction.

- Dans Elementor, entrez le nom dans le formulaire

Le nom du formulaire

Appeler une méthode Elementor

Elementor a prévu un **"crochet" (hook)** en PHP pour parvenir à extraire les données du formulaire et de réaliser une requête, par exemple. Celui-ci porte le nom de "**elementor_pro/forms/validation**". Mais commençons par le début.

- Créez un fichier dans votre thème enfant portant le nom de "**Travels_Query.php**" ou un nom à votre convenance, et glissez le dans le dossier "**php**".
- Créez dans votre fichier "**functions.php**", une nouvelle entrée.

```
// Travelsinclude_once ($dir . '/php/OP_Travels_Query.php');
```

- Dans le fichier "**Travels_Query.php**", créez le crochet ci-dessous.

```
<?php
  /**
  * Plugin Name: Travels_Query
  * Description: Travels archives filter
  * Author: Olivier Paudex
  * Author Web Site: https://www.fuyens.ch
  */
  function travels_filter($record, $ajax_handler) {
  }  add_action ('elementor_pro/forms/validation', 'travels_filter',10,2);
```

Voilà la base pour extraire les données d'un formulaire Elementor. Maintenant, si vous cliquez sur le bouton du formulaire, le code ci-dessus va s'exécuter.

Deux petites remarques sur les crochets en général :

- Le nom de la fonction "**travels_filter**" doit correspondre avec son appel, en bas dans la méthode " **add-action**".
- Les deux nombres correspondent à l'ordre d'exécution et aux nombres d'arguments (ici, il y en a deux).
- Dans la fonction PHP "**travels_filter**", entrez ces lignes. Ceci permet de vérifier que le code doit bien s'exécuter sur le formulaire "**filter_travels**".

```
// Check the Elementor form name
$form_name = $record->get_form_settings('form_name');
if ('filter_travels' !== $form_name) {
  return;}
```

Les listes déroulantes

Pour créer les listes déroulantes, il est bien entendu possible de les créer avec l'interface Elementor. Mais comme ces dernières ont tendances à se répéter, il est peut-être préférable de les créer une fois pour toute avec du code PHP.

Dans l'exemple ci-dessous, la fonction "**select_travels_type**" va vérifier que la demande vient bien du formulaire " **filter_travels**", puis que la dénomination du champ est bien "**travel_taxonomy**". SI c'est bien le cas, il va récupérer à l'aide de la fonction wordpress '**get_terms**", tous les termes du type voyages dont la taxonomie porte le libellé "**travel_type**". Et s'il n'y a pas d'erreurs, le code va ajouter en première position l'entrée " **Tous**", revenir à la ligne, insérez l'un après l'autre, tous les termes, sans ajouter de retour à la ligne pour le dernier.

```
function select_travels_type ($item, $index, $form) {
    // Check the Elementor form name
    if ('filter_travels' === $form->get_settings_for_display('form_name')) {

      // Check the Elementor field ID
      if ('travel_taxonomy' === $item['custom_id']) {
        // Get all the custom post type terms
        $terms = get_terms('travel_type');

        if (!empty($terms) && !is_wp_error($terms)) {

          // Add the placeholder item
          $item['field_options'] = "Tous" . "\n";

          // Add item to the selector
          foreach ($terms as $term) {
            $item['field_options'] .= $term->name;
            // Last one, don't add the "\n"
            if (!($term === end($terms))) {$item['field_options'] .= "\n";}
          }
        }
      }
    }
    return $item;
  }  add_filter ('elementor_pro/forms/render/item/select','select_travels_type',10,3);
```

Un peu plus simple à comprendre, le code qui gère l'affichage des mois de l'année.

```
// Populate Elementor selector form field with Travel months
  function populate_travels_elementor_dropdown_list ($item, $index, $form) {
    // Check the Elementor form name
    if ('filter_travels' === $form->get_settings_for_display('form_name')) {

      // Check the Elementor field ID
      if ('travel_month' === $item['custom_id']) {
        // Add the placeholder item
        $item['field_options'] = "Tous" . "\n";
        $item['field_options'] .= "Janvier" . "\n";
        $item['field_options'] .= "Février" . "\n";
        $item['field_options'] .= "Mars" . "\n";
        $item['field_options'] .= "Avril" . "\n";
        $item['field_options'] .= "Mai" . "\n";
        $item['field_options'] .= "Juin" . "\n";
        $item['field_options'] .= "Juillet" . "\n";
        $item['field_options'] .= "Août" . "\n";
        $item['field_options'] .= "Septembre" . "\n";
        $item['field_options'] .= "Octobre" . "\n";
        $item['field_options'] .= "Novembre" . "\n";
        $item['field_options'] .= "Décembre";
      }
    }
    return $item;
  }
  add_filter ('elementor_pro/forms/render/item/select','populate_travels_elementor_dropdown_lis
```

Dernière liste, celle du bouton radio "**ordre**". En effet, les boutons radios fonctionnent sur le même principe qu'une liste déroulante. Ajoutez le code ci-dessous. Celui-ci ressemble au code pour l'affichage des mois, à l'exception que le nom et sa valeur sont différents. Pour ajouter une valeur à une option, il faut ajouter la barre verticale, suivi de la donnée en question. Dans cette liste, pour l'option '**Croissant**", sa

valeur est "**asc**".

```
// Populate Elementor radio form field with ascending and descending order values
  function select_travels_order ($item, $index, $form) {
    // Check the Elementor form name
    if ('filter_travels' === $form->get_settings_for_display('form_name')) {

      // Check the Elementor field ID
      if ('travel_order' === $item['custom_id']) {
        // Add items to the radio selector
        $item['field_options'] = "Croissant" . "|asc" . "\n";
        $item['field_options'] .= "Décroissant" . "|desc";
      }
    }
    return $item;
  }  add_filter ('elementor_pro/forms/render/item/radio','select_travels_order',10,3);
```

Récupérer les données du formulaire

La prochaine étape consiste à récupérer les données du formulaire. Entrez les lignes de code ci-dessous à la suite dans la fonction PHP "**travels_filter**".

```
// Get the form datas
$fields = $record->get('fields');
$search_text = strtolower($fields['search_text']['value']);
$travel_country = strip_accents(strtolower($fields['travel_country']['value']));
$travel_month = strip_accents(strtolower($fields['travel_month']['value']));
$travel_year = $fields['travel_year']['value'];
$travel_taxonomy = strtolower($fields['travel_taxonomy']['value']);
$travel_order = strtolower($fields['travel_order']['value']);
// Get the terms slug
if ($travel_taxonomy != 'tous') {
  $travel_taxonomy = get_term_by('name', $travel_taxonomy, 'travel_type')->slug;}
```

Je fais l'impasse sur toute l'explication de ce code, mais la syntaxe est**$nom_de_variable = $fields['nom_du_champ']['value']**.

- La fonction WordPress "**get_term_by**" permet de récupérer les termes de la taxonomie selon un nom. Le **"slug"** en bout de chaîne est une syntaxe propre à la POO (programmation objet), qui récupère le slug au lieu du nom du terme. Ce qui nous donne l'assurance que celui-ci est écrit en minuscule et sans espace.
- La fonction PHP "**strlower**" permet de retourner les caractères en minuscule.
- Quant aux "**strip_accents**", c'est une fonction sortie de ma trousse à outil qui supprime tous les caractères accentués. La voici :

```
function strip_accents($str) {
  return strtr(utf8_decode($str), utf8_decode('àáâãäçèéêëìíîïñòóôõöùúûüýÿÀÁÂÃÄÇÈÉÊËÌÍÎÏÑÒÓÔÕÖÙ
}
```

Redirection vers une nouvelle page

Il ne reste plus qu'à créer une URL avec toutes les données récupérées, exécuter la requête sur la base de données et afficher le résultat. Mais avant de le coder, petit rappel de principe sur les URL dans WordPress :

- Pour afficher la page d'accueil, entrez **"https://www.fuyens.ch**".
- Pour afficher une page d'archives dont le type personnalisé est "**voyages**", entrez " **https://www.fuyens.ch/voyages**".
- Pour afficher une page d'archives dont le type personnalisé est "**voyages**" et la chaîne de recherche est " **suisse**", entrez " **https://www.fuyens.ch/voyages/?s=suisse**".
- Une autre syntaxe également valable est celle-ci : " **https://www.fuyens.ch/?post_type=voyages&s=suisse**".
- Pour afficher une page d'archives dont la chaîne de recherche est "**bern**" et le nom du pays est " **suisse**", entrez " **https://www.fuyens.ch/voyages/?s=bern&travel_country=suisse**".

> Attention, si vous entrez la syntaxe **https://www.fuyens.ch/?s=suisse**, sans préciser le type de publications, une page de résultats de recherche va s'ouvrir. On ne va pas parler de la page de résultats de la recherche maintenant, mais sachez qu'il est possible d'en créer une avec Elementor, au même titre qu'une page d'archives.

Une fois la syntaxe URL acquise, il est facile de compléter notre code.

- On commence par l'affichage de "**https://www.fuyens.ch/"** avec la fonction "**home_url()**".
- Puis, on ajoute le mot "**voyages**", suivi d'un slash (**/**).
- Puis, on ajoute la syntaxe **"?s="** et on y ajoute le contenu de la variable "**$search_text**".
- Puis, on recommence avec tous les autres paramètres de notre formulaire de recherche.

Encore quelques infos.

- La fonction PHP "**rawurldecode**" permet de remplacer les caractères de formatage.

- La syntaxe spéciale utilisé ici n'est rien d'autre qu'un "**if – else**" en version raccourci, ce qui facilite la lecture.
- La notion de **".="** permet de concaténer une chaîne de caractère.
- A la fin du code, se trouve la méthode Elementor **"add_response_data"**, qui permet d'envoyer notre variable "**$redirect_url**" au navigateur.

```
$redirect_url = home_url('voyages' . '/');
$search_text ? $redirect_url .= rawurldecode('?s=' . $search_text) : $redirect_url .= rawurldecode('?s=');
$travel_country ? $redirect_url .= rawurldecode('&travel_country=' . $travel_country) : $redirect_url .= rawurldecode('&travel_country=');
$travel_month ? $redirect_url .= rawurldecode('&travel_month=' . $travel_month) : $redirect_url .= rawurldecode('&travel_month=');
$travel_year ? $redirect_url .= rawurldecode('&travel_year=' . $travel_year) : $redirect_url .= rawurldecode('&travel_year=');
$travel_taxonomy ? $redirect_url .= rawurldecode('&travel_taxonomy=' . $travel_taxonomy) : $redirect_url .= rawurldecode('&travel_taxonomy=');
$travel_order ? $redirect_url .= rawurldecode('&travel_order=' . $travel_order) : $redirect_url .= rawurldecode('&travel_order=');
// Add the redirect to ajax handler and force reload
$ajax_handler->add_response_data('redirect_url', $redirect_url);
```

Et le code complet de la fonction "**travels_filter**".

```
<?php
  /**
  * Plugin Name: Travels_Query
  * Description: Travels archives filter
  * Author: Olivier Paudex
  * Author Web Site: https://www.fuyens.ch
  */
  function travels_filter($record, $ajax_handler) {
    // Check the Elementor form name
   $form_name = $record->get_form_settings('form_name');
   if ('filter_travels' !== $form_name) {
      return;
    }
    // Get the form datas
    $fields = $record->get('fields');
    $search_text = strtolower($fields['search_text']['value']);
    $travel_country = strip_accents(strtolower($fields['travel_country']['value']));
    $travel_month = strip_accents(strtolower($fields['travel_month']['value']));
    $travel_year = $fields['travel_year']['value'];
    $travel_taxonomy = strtolower($fields['travel_taxonomy']['value']);
    $travel_order = strtolower($fields['travel_order']['value']);
    // Get the terms slug
    if ($travel_taxonomy != 'tous') {
      $travel_taxonomy = get_term_by('name', $travel_taxonomy, 'travel_type')->slug;
    }

    $redirect_url = home_url('voyages' . '/');
    $search_text ? $redirect_url .= rawurldecode('?s=' . $search_text) : $redirect_url .= rawurldecode('?s=');
    $travel_country ? $redirect_url .= rawurldecode('&travel_country=' . $travel_country) : $redirect_url .= rawurldecode('&travel_country=');
    $travel_month ? $redirect_url .= rawurldecode('&travel_month=' . $travel_month) : $redirect_url .= rawurldecode('&travel_month=');
    $travel_year ? $redirect_url .= rawurldecode('&travel_year=' . $travel_year) : $redirect_url .= rawurldecode('&travel_year=');
    $travel_taxonomy ? $redirect_url .= rawurldecode('&travel_taxonomy=' . $travel_taxonomy) : $redirect_url .= rawurldecode('&travel_taxonomy=');
    $travel_order ? $redirect_url .= rawurldecode('&travel_order=' . $travel_order) : $redirect_url .= rawurldecode('&travel_order=');
    // Add the redirect to ajax handler and force reload
    $ajax_handler->add_response_data('redirect_url', $redirect_url);
  }
  add_action ('elementor_pro/forms/validation', 'travels_filter',10,2);?>
```

Voilà, si tout se déroule normalement, votre navigateur devrait vous afficher **une jolie page 404**, avec, dans la barre URL, la sortie du formulaire.

Mais, bien entendu, ceci n'est pas le résultat attendu. Le navigateur doit maintenant savoir envoyer cette requête à la base de données de WordPress et afficher les bonnes publications selon les critères de recherche, dans la page d'archives.

A suivre dans le prochain chapitre "**Générer une requête SQL sur WordPress**".