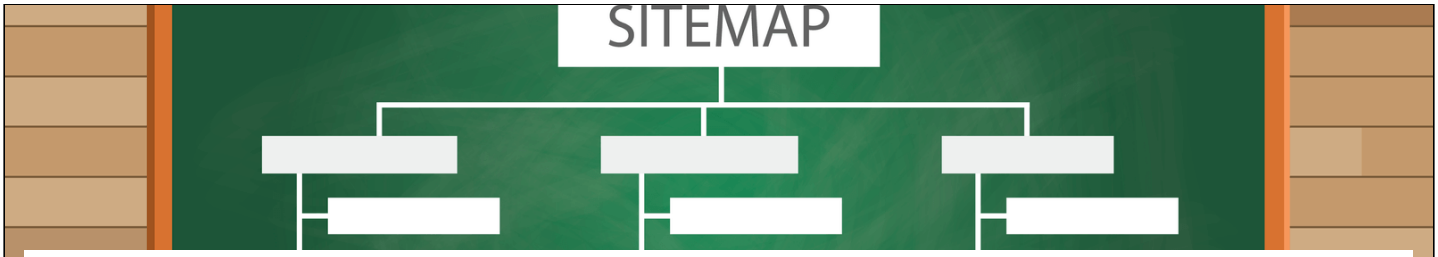


Pourquoi créer un plan du site (sitemap)

Elementor, PHP, Polylang

Web Compétences : 4

Avez-vous déjà entendu parler de sitemap dans une conversation sans jamais vraiment comprendre de quoi il en retourne ? Savez-vous qu'il existe deux sortes de sitemap ? Le premier n'est rien d'autre que le plan du site ou le plan de toutes les pages publiées. Le deuxième, beaucoup plus subtile, est une sorte d'index réservé aux moteurs de référencement. Alors, comment bien construire un sitemap ? Voici quelques astuces glaner ici et là pour la construction de ce site.

Publié jeudi 24 juin 2021, 10h45
Modifié lundi 26 août 2024, 10h04

By Olivier Paudex

Introduction

Un sitemap a deux fonctions primaires. La première est le plan du site à proprement parlé, qui permet aux visiteurs de consulter le catalogue de toutes les pages et publications. La deuxième est son indexation sous forme de fichier XML, qui permet aux moteurs de recherche comme Google de vous retrouver et de référencer votre siteweb. Cependant, le sitemap ne remplacera jamais les bonnes pratiques du SEO que sont le schéma, le titre et la méta-description, entre autres. Voici une petite réflexion personnelle qui va énumérer les différences entre les deux sortes de sitemap, les étapes importantes lors de la création ainsi que les différentes méthodes pour les réaliser.

Indexer ou positionner son site web

Beaucoup d'encre a coulé sur ce sujet houleux. Faut-il obligatoirement créer un sitemap pour indexer ses pages ou positionner son site web ? La réponse est **2x NON**. Un site web bien conçu se verra indexer automatiquement par Google et ses principaux concurrents. La raison du sitemap est l'accélération de la procédure par **un facteur de 100**. Oui, vous avez bien lu. Une nouvelle publication se verra indexer par Google après **environ 15 minutes**. Il faut compter **24 heures** pour que ce dernier le fasse sans sitemap. Il faut néanmoins que le sitemap soit valide et que celui-ci soit référencé à l'aide de la console **"Google Search"**.

Quant au positionnement du site, rien à voir avec le sitemap, mais alors rien. Tous les spécialistes en SEO vous le diront. Les secrets d'un bon positionnement sont :

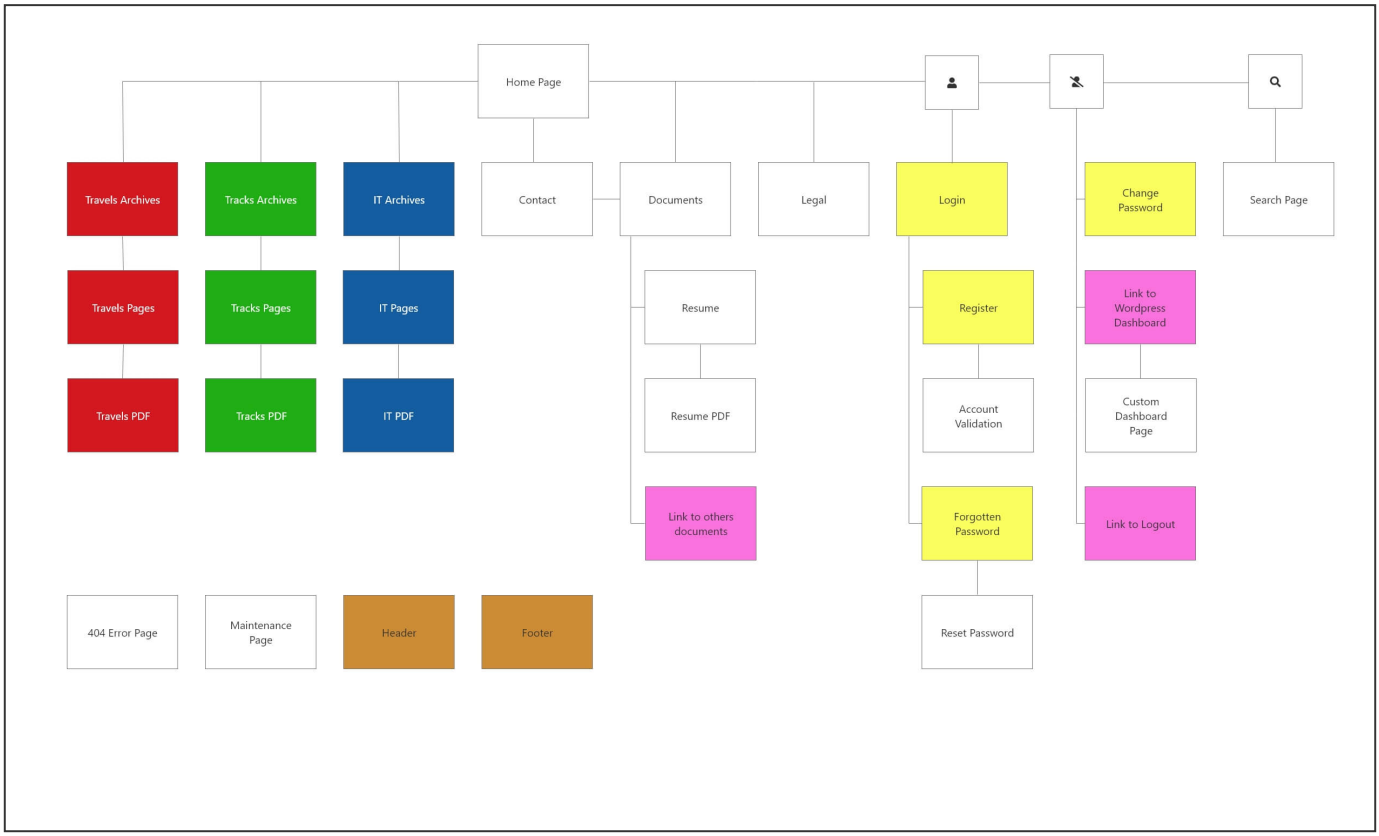
- L'utilisation d'un schéma hiérarchique.
- L'utilisation de mots clés judicieux sur vos publications.
- Un titre pertinent.
- Une description de votre publication décrivant en quelques mots le contenu de ce qui va suivre.
- Une hiérarchie dans les urls et la réécriture de ceux-ci si nécessaire.

Vitesse et tampon mémoire

Vous l'aurez bien compris, le sitemap accélère l'indexation du site lors de nouvelles publications. Mais le temps gagné à l'aide d'un sitemap n'est pas seulement une affaire d'indexation. Une autre bonne idée pour accélérer l'affichage de son site web est l'utilisation d'un plugin de mise en tampon mémoire, plus communément appelé **"cache"**. Le cache n'est rien d'autre qu'une copie du code HTML généré, mais de façon statique. A l'aide d'un fichier sitemap qui contient les urls de toutes les publications du site, on peut **"précharger"** dans le cache, tout le contenu du site à une intervalle de temps régulière. Ce site utilise le plugin **"WP-Rocket"** qui se base sur le sitemap du site pour créer son cache.

Le sitemap au format HTML

Le premier sitemap n'est rien d'autre que le plan de toutes les pages, de tous les termes rencontrés dans les différentes taxonomies et d'autres informations optionnelles. Pour ce site, la réalisation du plan est basée sur l'étude de départ et utilise le widget natif d'Elementor.



Plan du site réalisé lors de la phase de conception

- La plupart des sitemaps ne font que lister les publications et certaines pages. Tout le reste, comme les pages d'archives, les PDF et le système de connexion n'a pas été pris en compte.
- Les termes de taxonomies, qui n'apparaissent pas sur la conception ci-dessus, ont, par contre été ajoutés au sitemap.
- Le lien du sitemap est souvent situé dans le pied de page. **La page a été rebaptisé site-map (en deux mots)**, pour éviter la confusion avec son homologue, l'index XML.

Créer le plan du site avec Elementor

Voici à quoi ressemble le sitemap de **fuyens.ch**.

Elementor fourni un widget pour créer le sitemap. Si celui-ci est relativement complet, il est tout a fait envisageable de le modifier pour agir sur l'affichage des données.

Le widget sitemap

Pour créer un sitemap, il faut bien entendu commencer par créer une nouvelle page, puis d'y glisser le widget sitemap dans une section. Celui-ci se présente comme le widget de la liste d'icônes, à savoir, une suite d'éléments.

Il est bien entendu possible de modifier les styles comme la couleur et la typographie et le nombre de colonnes à l'affichage.

Néanmoins, le sitemap est assez basique et ne permet pas, par exemple de créer son propre titre et de modifier sa couleur (une couleur par type de publications, par exemple). Comme souvent avec Elementor, il faut recourir au CSS pour y parvenir. Voici quelques astuces.

- Pour supprimer un titre, la classe utilisée par Elementor est **"elementor-sitemap-CPT-title"**, où il faut remplacer **"CPT"** par le type de publications.
- Pour supprimer le titre d'une taxonomie, c'est la même chose.
- La propriété CSS à utiliser est **"display: none"**.

```
.elementor-sitemap-it-title, .elementor-sitemap-travels-title, .elementor-sitemap-tracks-title,
.elementor-sitemap-it_type-title, .elementor-sitemap-travel_type-title, .elementor-sitemap-cycle_type-title {
  display: none;}
```

- Pour gérer les listes, Elementor utilise les puces, mais a complétement omis l'espacement entre les lignes. Là aussi, CSS vient à la rescousse avec la classe **".elementor-sitemap-item a"**.
- Pour gérer la sélection au passage de la souris, on peut utiliser la propriété **"hover"** de la même classe que ci-dessus.

```
.elementor-sitemap-item a {
  color: #333333;
  line-height: 40px;
}
.elementor-sitemap-item a:hover {
  color:#2E8BC0;
  text-decoration: underline;}
```

- Pour créer le titre et le compteur de publications, il faut utiliser le widget **"Titre"**.
- Pour créer le compteur, il faut utiliser le paramètre dynamique **"nombre de publications"** et le régler pour afficher le bon type de publications.
- Pour créer le compteur de termes d'une taxonomie, il n'y a malheureusement pas de paramètre dynamique disponible. Il faut donc utiliser le paramètre **"code court"** à la place.

Ci-dessous, un exemple de code court pour afficher le nombre de termes du type personnalisé **"it_type"**. Le paramètre **"hide_empty"** permet de n'afficher que les termes pour lesquels une publication existe.

```
// IT Terms Counter
function it_terms_counter() {
  return wp_count_terms(array('taxonomy' => 'it_type', 'hide_empty' => true));
}add_shortcode ('IT_Terms_Counter','it_terms_counter');
```

Avec un peu de CSS, il est tout a fait possible de créer de magnifiques plan de site. Il est donc inutile d'utiliser un autre plugin que celui fourni par Elementor.

Le sitemap au format XML

Le deuxième sitemap est beaucoup plus compliqué à gérer. Il s'agit de créer et de gérer un fichier XML contenant toutes les pages, les taxonomies et autres informations que le site voudrait transmettre aux moteurs de recherche. Pour accéder à un sitemap, il faut entrer l'URL ci-dessous :

https://www.fuyens.ch/sitemap.xml

Bien entendu, il faut remplacer le nom de domaine par le sien.

Méthode automatique avec un plugin

La première méthode pour créer un sitemap est d'utiliser un plugin comme **Yoast** ou **Rank Math**. Tous ces plugins de SEO permettent de fabriquer un sitemap plus ou moins automatiquement. Ils ont tous des avantages et des inconvénients. Ce guide ne va pas rentrer en matière pour comparer le meilleur plugin de sitemap, mais plutôt se pencher sur les méthodes alternatives.

Méthode automatique sans plugin

Une des autres méthodes pour créer un sitemap est de le faire directement avec WordPress. En effet, WordPress a intégré cette fonctionnalité depuis la version 5.5.

Pour activer le sitemap automatique, il faut ajouter cette ligne de PHP dans le fichier **"functions.php"**.

```
// Enable sitemapadd_filter('wp_sitemaps_enabled', '__return_true');
```

Supprimer les modèles Elementor

Cette méthode entièrement automatique permet de créer un sitemap complet, trop complet. En effet, avec Elementor, les types de modèles sont considérés comme des pages. Il faut alors les supprimer avec le code ci-dessous.

```
// Remove elementor library templates from WP Sitemap
function remove_elementor_templates_from_wp_sitemap ($post_types) {
  unset ($post_types['elementor_library']);
  return $post_types;
}add_filter('wp_sitemaps_post_types', 'remove_elementor_templates_from_wp_sitemap');
```

Supprimer les utilisateurs

Dans le même ordre d'idées, il est possible de filtrer et de supprimer les utilisateurs qui possède un compte sur le site. C'est le genre d'informations qui n'a aucune raison d'être indexé par Google.

```
// Remove users pages from sitemap
function remove_users_from_wp_sitemap ($provider, string $name) {
  if ($name == 'users') {
    return false;
  }
  return $provider;
}add_filter('wp_sitemaps_add_provider', 'remove_users_from_wp_sitemap', 10, 2);
```

Supprimer une taxonomie

Si l'on désire supprimer la taxonomie du type personnalisé **"IT"**, c'est le même code...ou presque.

```
function remove_tax_from_wp_sitemap ($tax) {
  unset ($tax['it_type']);
  return $tax;
}add_filter('wp_sitemaps_taxonomies', 'remove_tax_from_wp_sitemap');
```

Toute l'information officiel de WordPress se trouve **ici**.

Ajoutez un provider

C'est quoi un provider ? Selon la documentation de WordPress, un provider n'est rien d'autre qu'un type de contenu. Une page ou un article est un type de contenu. Une image est aussi un type de contenu bien à part. Les types personnalisés, qui sont des articles avec d'autres meta-données sont également d'autres types de contenu. Un type de contenu dont personne ne pense en premier, sont les urls. En effet, les urls qui ne représentent pas des pages ou des articles, mais dans le cas d'Elementor, une page d'archive ou une page de résultat de recherche, ne sont pas automatiquement ajoutées au sitemap. Dans ces cas là, il faut créer un nouveau provider pour pouvoir ajouter un nouveau type de contenu.

Ci-dessous, un exemple de nouveau provider pour créer des urls personnalisés.

```
// Add custom sitemap provider
function add_custom_sitemap_provider() {

  // Call an instance of the class WP_Custom_Sitemaps
  $name = 'customurls';
  $provider = new WP_Custom_Sitemaps($name);
  wp_register_sitemap_provider($name, $provider);
}add_filter ('init', 'add_custom_sitemap_provider');
```

Une fois l'instance **"WP_Custom_Sitemaps"** créé, il faut étendre la classe **"WP_Sitemaps_Provider"** et y ajouter son propre code. A savoir que la classe de base contient deux fonctions publiques**"get_url_list"**, qui permet d'afficher ses urls personnalisés et **"get_max_num_pages"** qui permet d'afficher le nombre de pages. Pour faire simple, l'exemple ci-dessous se base sur une page unique.

```
class WP_Custom_Sitemaps extends WP_Sitemaps_Provider {
  // Init
  public function __construct($name) {
    $this->name        = $name;
    $this->object_type = 'post';
  }
  // Get the URL list
  public function get_url_list($page, $post_type = '') {
    $urls = [];
    $urls[] = array(
      'loc' => 'https://staging.fuyens.ch/fr/voyages/'
    );
    return $urls;
  }
  // Get the number of pages
  public function get_max_num_pages($post_type = '') {
    return 1;
  }}
```

Méthode manuelle

La méthode WordPress ci-dessus est bien pratique. Malgré ceci, il est aussi possible de créer son propre sitemap à la main, en partant d'une feuille blanche. La première raison de le faire est certainement la mauvaise gestion du multilinguisme.

Ci-dessous, les étapes pour y parvenir.

Réécriture des urls

Premièrement, il faut réécrire l'URL pour accéder à votre (vos) sitemaps.

- Le premier **"rewrite_rule"** va appeler l'index du sitemap quand on entre l'url **"https://www.fuyens.ch/sitemap.xml/"**.
- Le deuxième **"rewrite_rule"** va appeler le sitemap correspondant en fonction du type de contenu, ainsi que la page si le nombre de publications dépassent les 2000. Par exemple, l'url **"https://www.fuyens.ch/fr/sitemap-posts-it-1.xml/"** va afficher la première page du sitemap concernant les publications "IT".

```
<?php
  /**
  * Plugin Name: OP_XML_Sitemap
  * Description: Create an XML sitemap
  * Author: Olivier Paudex
  * Author Web Site: https://www.fuyens.ch
  */
  // Init
  function add_custom_sitemap() {

    // Disable Wordpress default sitemap
    add_filter('wp_sitemaps_enabled', '__return_false');
    // Constant number of posts per page
    define("NUMBER_OF_POSTS_PER_PAGE", 2000);
    // Add the new custom tag
    add_rewrite_tag('%template%', '([^?]+)');
    add_rewrite_tag('%subtype%', '([^?]+)');
    // Flush the rules
    flush_rewrite_rules();
    // Rewrite rules
    add_rewrite_rule('^sitemap\.xml$', 'index.php?template=index', 'top');
    add_rewrite_rule('^/sitemap-([a-z]+)-([a-z\d_-]+)-(\d+)\.xml$',
      'index.php?template=$matches[1]&subtype=$matches[2]&paged=$matches[3]',
      'top'
    );
  }  add_filter ('init', 'add_custom_sitemap');
```

Afficher le sitemap

La deuxième étape consiste à afficher le sitemap désirée grâce à la fonctionnalité **"template_redirect"**.

```
// Render sitemaps
function render_sitemaps() {
  // Get the value of the custom tag
  $template = sanitize_text_field (get_query_var('template'));
  $subtype  = sanitize_text_field(get_query_var('subtype'));
  $paged    = absint(get_query_var('paged'));
  // Exclude posts or pages
  $pages = array('pdf-travels', 'pdf-tracks', 'pdf-it', 'pdf-cv-fr', 'pdf-cv-en', 'reset-passwo
  $exclude = exclude_pages ($pages);

  // Render the sitemap
  if (!empty($template)) {
    if ($template === 'index') echo render_index($exclude);
    if ($template === 'posts' && !empty($subtype)) echo render_posts($subtype, $exclude, $paged
    if ($template === 'taxonomies' && !empty($subtype)) echo render_terms($subtype, $paged);
    exit;
  }
}add_action ('template_redirect', 'render_sitemaps');
```

La fonction qui permet d'exclure des pages est ci-dessous. Elle utilise les fonctions de Polylang pour retrouver une publication et sa traduction. Ainsi, il n'est pas nécessaire de faire la liste de toutes les publications à exclure. Seule une suffit.

```
// Return an array of pages ID to exclude in all languages
function exclude_pages ($pages) {
  // Get all languages
  if (function_exists('pll_languages_list')) {$lists = pll_languages_list();}

  // Get ID of the pages
  foreach ($pages as $page) {
    if (function_exists('pll_get_post')) {
      foreach ($lists as $list) {
        $pageID = pll_get_post(get_page_by_path($page)->ID, $list);
        if (!empty($pageID)) {
          $exclude[] = $pageID;
        }
      }
    } else {
      $pageID = get_page_by_path($page)->ID;
      $exclude[] = $pageID;
    }
  }

  return $exclude;}
```

Afficher l'index du sitemap

La troisième étape va consister à afficher l'index. Voici quelques explications.

- Les premières lignes concernent la mise en place d'un fichier XSL.
- Les suivantes mettent en place la structure du fichier XML avec son header.
- Le groupe des **"unset"** permettent de supprimer les types non désirés. Sur cet exemple, les modèles "Elementor", le type article et les attachements ne sont pas pris en compte.
- Ensuite, pour chaque langue, chaque type de publications et chaque taxonomie, le nombre de pages est calculé et les liens de l'index sont créés en utilisant la balise**"<loc>"**.

```
// Render the sitemap index
function render_index($exclude) {
  // Path to XSL file
  $protocol = (isset($_SERVER['HTTPS']) && $_SERVER['HTTPS'] === 'on' ? "https://" : "http://");
  $homeurl = $protocol . $_SERVER['SERVER_NAME'];
  $xslpath = $homeurl . '/wp-content/themes/hello-theme-child-master/sitemap/';

  // Header
  header("Content-type: text/xml");
  $xml = '<?xml version="1.0" encoding="UTF-8"?>';
  $xml .= '<?xml-stylesheet type="text/xsl" href="' . $xslpath . 'sitemap.xsl' . '"?>';
  $xml .= '<sitemapindex xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">';
  // Get all public post types excluding elementor templates, posts and attachments
  $args = array('public' => true);
  $post_types = get_post_types($args);
  unset ($post_types['elementor_library']);
  unset ($post_types['post']);
  unset ($post_types['attachment']);
  // Get all languages
  if (function_exists('pll_languages_list')) {$lists = pll_languages_list();}
  // For each language
  foreach ($lists as $list) {
    // Create the post types index
    if ($post_types) {
      foreach ($post_types as $post_type) {
        // Count number of required pages
        $count_posts = count(get_filtered_posts ($post_type, $exclude, -1, $paged, $list));
        $required_paged = ceil($count_posts/NUMBER_OF_POSTS_PER_PAGE);

        for ($paged = 1; $paged <= $required_paged; $paged++) {
          $xml .= '<sitemap>';
          $xml .= '<loc>' . home_url($list . '/' . 'sitemap-posts-' . $post_type . '-' . $paged
          $xml .= '</sitemap>';
        }
        // Get all taxonomies for each post type
        $args = array('public' => true, 'object_type' => array($post_type));
        $taxonomies = get_taxonomies($args);
        // Create the taxonomies index
        if ($taxonomies) {
          foreach ($taxonomies as $taxonomy) {
            // Count number of required pages
            $count_terms = wp_count_terms($taxonomy, array('hide_empty'=> true));
            $required_paged = ceil($count_terms/NUMBER_OF_POSTS_PER_PAGE);

            for ($paged = 1; $paged <= $required_paged; $paged++) {
              $xml .= '<sitemap>';
              $xml .= '<loc>' . home_url($list . '/sitemap-taxonomies-' . $taxonomy . '-' . $pa
              $xml .= '</sitemap>';
            }
          }
```

```
        }
      }
    }
  }

  // Sitemap Footer
  $xml .= '</sitemapindex>';
  return $xml;}
```

La fonction **"get_filtered_posts"** est décrite ci-dessous. Elle utilise la fonction WordPress **"get_posts"**.

```
// Get list of posts or pages
function get_filtered_posts ($subtype, $exclude, $numberposts, $paged, $lang) {
  $posts = get_posts(array('post_type' => $subtype, 'exclude' => $exclude, 'numberposts' => $numberposts,
                           'paged' => $paged, 'lang' => $lang, 'post_status' => 'publish'));
  return $posts;}
```

Afficher les publications du sitemap

Le code ressemble à celui de l'index, à l'exception prêt qu'il est possible de gérer les différentes langues à l'aide des balises **"<xhtml>"** et donner sa traduction avec la balise **"alternate"**. Bien entendu, il faut installer le plugin Polylang, qui gère ceci à la perfection.

De plus, il ajoute la date et l'heure de la dernière modification de la publication avec la balise**"<lastmod>"**.

```
// Render the sitemap posts
function render_posts($subtype, $exclude, $paged) {
  // Path to XSL file
  $protocol = (isset($_SERVER['HTTPS']) && $_SERVER['HTTPS'] === 'on' ? "https://" : "http://");
  $homeurl = $protocol . $_SERVER['SERVER_NAME'];
  $xslpath = $homeurl . '/wp-content/themes/hello-theme-child-master/sitemap/';
  // Header
  header("Content-type: text/xml");
  $xml = '<?xml version="1.0" encoding="UTF-8"?>';
  $xml .= '<?xml-stylesheet type="text/xsl" href="' . $xslpath . 'sitemap.xsl' . '"?>';
  // Urlset Header
  $xml .= '<urlset ';
  $xml .= 'xmlns:image="http://www.google.com/schemas/sitemap-image/1.1" ';
  $xml .= 'xmlns:xhtml="http://www.w3.org/1999/xhtml" ';
  $xml .= 'xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">';
  // Get all languages
  if (function_exists('pll_languages_list')) {$lists = pll_languages_list();}
  // Get posts or pages
  $posts = get_filtered_posts ($subtype, $exclude, NUMBER_OF_POSTS_PER_PAGE, $paged, $lang);
  // Create the posts or pages url
  if ($posts) {
    foreach ($posts as $post) {
      $xml .= '<url>';
      $xml .= '<loc>' . get_permalink($post->ID) . '</loc>';
      $xml .= '<lastmod>'. get_the_modified_date('Y-m-d\TH:i:sP', $post) .'</lastmod>';
      // Get other language page from Polylang
      foreach ($lists as $list) {
          if (function_exists('pll_get_post')) {
            $postID = pll_get_post($post->ID, $list);
            if ($postID) {
              $xml .= '<xhtml:link hreflang="' . $list . '" href="' . get_permalink($postID) .
```

```
            }
          }
        }

        // Footer
        $xml .= '</url>';
      }
    }
    // Urlset Footer
    $xml .= '</urlset>';
    return $xml;  }
```

Afficher les termes du sitemap

Pour chaque taxonomie, le sitemap va afficher les termes lui correspondant. Le code est identique à celui des publications.

```
// Render the sitemap terms
function render_terms($subtype, $paged) {
  // Path to XSL file
  $protocol = (isset($_SERVER['HTTPS']) && $_SERVER['HTTPS'] === 'on' ? "https://" : "http://");
  $homeurl = $protocol . $_SERVER['SERVER_NAME'];
  $xslpath = $homeurl . '/wp-content/themes/hello-theme-child-master/sitemap/';
  // Header
  header("Content-type: text/xml");
  $xml = '<?xml version="1.0" encoding="UTF-8"?>';
  $xml .= '<?xml-stylesheet type="text/xsl" href="' . $xslpath . 'sitemap.xsl' . '"?>';
  // Urlset Header
  $xml .= '<urlset ';
  $xml .= 'xmlns:image="http://www.google.com/schemas/sitemap-image/1.1" ';
  $xml .= 'xmlns:xhtml="http://www.w3.org/1999/xhtml" ';
  $xml .= 'xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">';
  // Get all languages
  if (function_exists('pll_languages_list')) {$lists = pll_languages_list();}
  // Get terms
  $terms = get_terms(array('taxonomy' => $subtype, 'hide_empty' => true, 'number' => NUMBER_OF_POSTS_PE
  'offset' => (($paged * NUMBER_OF_POSTS_PER_PAGE) - NUMBER_OF_POSTS_PER_PAGE)));
  // Create the terms url
  if ($terms) {
    foreach ($terms as $term) {
      $xml .= '<url>';
      $xml .= '<loc>' . get_term_link($term) . '</loc>';

      // Get other language page from Polylang
      foreach ($lists as $list) {
        if (function_exists('pll_get_term')) {
          $termID = pll_get_term($term->term_id, $list);
          if ($termID) {
            $xml .= '<xhtml:link hreflang="' . $list . '" href="' . get_term_link($termID) . '" rel="a
          }
        }
      }
      $xml .= '</url>';
    }
  }

  // Urlset Footer
  $xml .= '</urlset>';

  return $xml;}
```

La feuille de style pour le sitemap

Si le HTML a le CSS pour personnalisé son style, le XML a lui aussi sa feuille de style. Elle porte l'extension XSL pour **"Extended Stylesheet Language"**. Grâce à elle, on peut rendre notre feuille de style très jolie. (ce qui dans le cas d'un fichier sitemap, ne sert pas à grand chose).

La mise en beauté du sitemap de ce site a été réalisé par Pedro Borges. Vous pouvez retrouver la feuille de style sur son **github**.

Le fichier XSL téléchargé, il faut le copier dans votre thème enfant et le référencer dans le code. Les lignes de code ci-dessous vont

- Vérifier si votre site utilise le protocole **"HTTP"** ou **"HTTPS"**.
- Récupérer l'url de votre nom de domaine.
- Récupérer l'url du fichier xsl, qui est situé, pour cet exemple, dans le dossier**"sitemap"** du thème enfant **"hello"** d'Elementor.

```
// Path to XSL file
$protocol = (isset($_SERVER['HTTPS']) && $_SERVER['HTTPS'] === 'on' ? "https://" : "http://");
$homeurl = $protocol . $_SERVER['SERVER_NAME'];
$xslpath = $homeurl . '/wp-content/themes/hello-theme-child-master/sitemap/';
```

Conclusions

Créer un sitemap pour son site web est optionnel, mais chaudement recommandé. D'une part, un sitemap au format HTML permet à tout un chacun de retrouver une publication ou de voir le contenu d'un site. On peut utiliser la recherche du navigateur **"CTRL+F"**, sur la page du sitemap, ce qui va bien plus vite que n'importe quel moteur de recherche. Le fichier XML, quant à lui, permet d'accélerer l'indexation des pages chez Google, mais n'apportera aucun bénéfice sur le classement (ranking). Par contre, le sitemap est obligatoire avec la plupart des plugins de cache comme **"Wp-Rocket"** ou **"Wp-Optimize"**. L'utilisation de ces derniers rend le sitemap quasi indispensable, si l'on désire afficher les pages de son site rapidement.