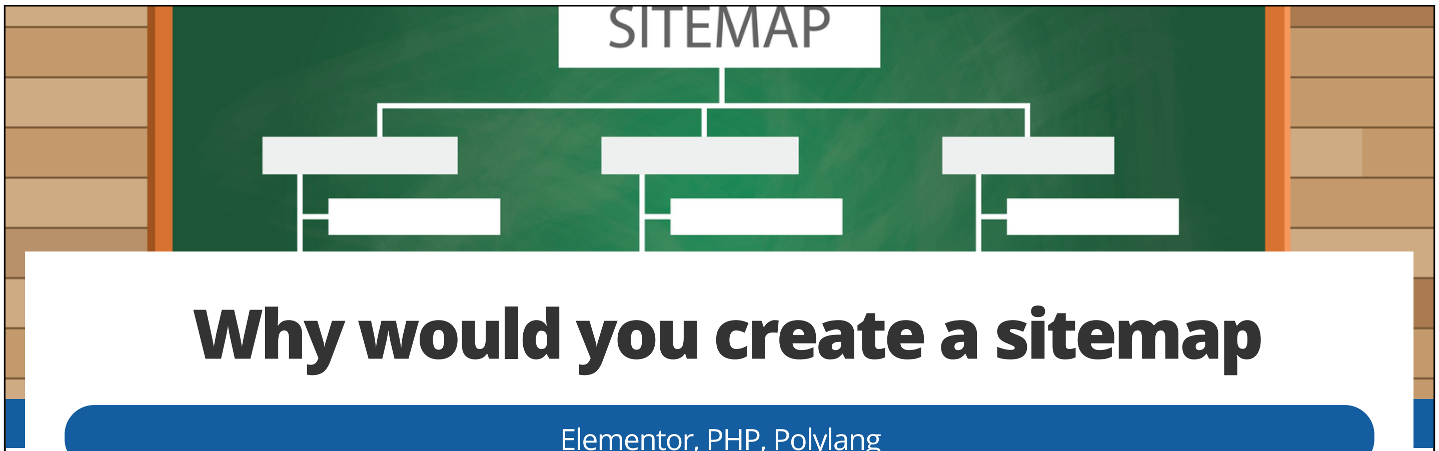


Why would you create a sitemap

Elementor, PHP, Polylang

Web Skills : 4

Have you ever heard about sitemap in a conversation without really understanding what it is about? Do you know that there are two kinds of sitemap? The first one is nothing else than the map of all published pages. The second one, much more subtle, is a kind of index reserved for referencing engines. So, how to build a sitemap ? Here are some tips gleaned here and there for the construction of this site.

Published Thursday June 24th 2021, 10:45
Modified Monday August 26th 2024, 10:04

By Olivier Paudex

Introduction

A sitemap has two primary functions. The first is the sitemap itself, which allows visitors to consult the catalog of all pages and publications. The second one is its indexation in the form of an XML file, which allows search engines like Google to find you and to reference your website. However, the sitemap will never replace the good practices of SEO that are the schema, the title and the meta-description, among others. Here is a little personal reflection that will list the differences between the two kinds of sitemap, the important steps during the creation as well as the different methods to achieve them.

Indexing or positioning your website

Much ink flowed on this stormy subject. Is it necessary to create a sitemap to index its pages or position its website? The answer is **2x NO**. A well designed website will be automatically indexed by Google and its main competitors. The reason of the sitemap is the acceleration of the procedure by**a factor of 100**. Yes, you read well. A new publication will be indexed by Google after about**15 minutes**. It takes **24 hours** for it to do so without a sitemap. However, it is necessary that the sitemap is valid and that it is referenced using the console **"Google Search"**.

As for the positioning of the site, nothing to do with the sitemap, really nothing. All SEO specialists will tell you. The secrets of a good positioning are:

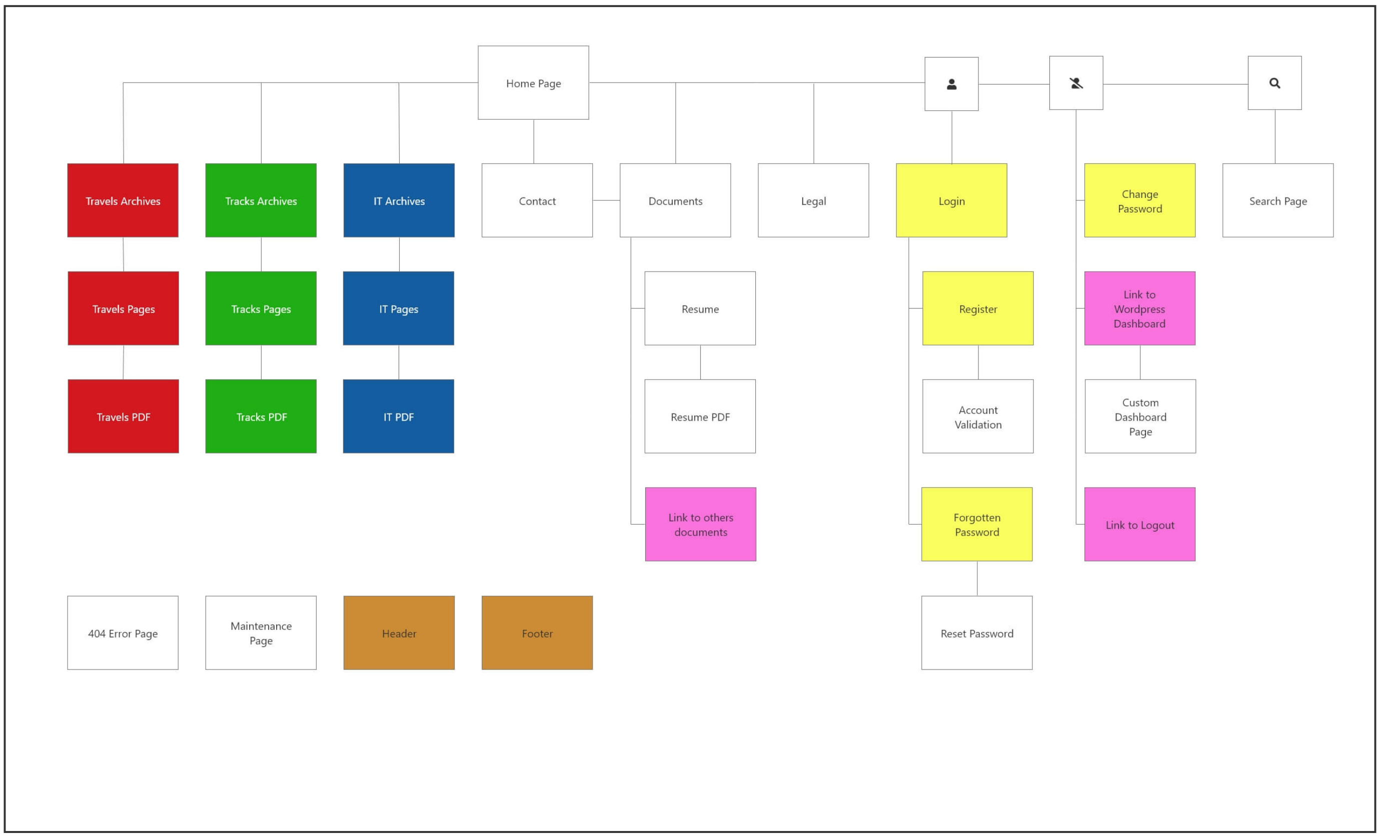
- The use of a hierarchical scheme
- The use of judicious keywords on your posts
- A relevant title
- A description of your post describing in a few words the content of what's to come.
- A hierarchy in the urls and rewriting them if necessary.

Speed and memory cache

As you can see, the sitemap speeds up the indexing of the site during new publications. But the time saved using a sitemap is not just about indexing. Another good idea to speed up the display of the website is the use of a memory buffering plugin, more commonly called **"cache"**. The cache is nothing more than a copy of the generated HTML code, but in a static way. Using a sitemap file that contains the urls of all the publications of the site, we can **"preload"** in the cache, all the content of the site at a regular time interval. This site uses the plugin **"WP-Rocket"** which is based on the sitemap of the site to create its cache.

The sitemap in HTML format

The first sitemap is nothing more than a map of all the pages, of all the terms encountered in the various taxonomies, and other optional informations. For this site, the creation of the sitemap is based on the initial study and uses the native Elementor widget.



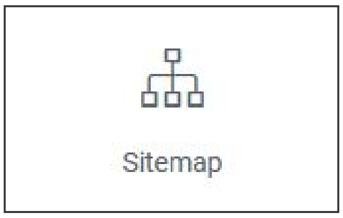
Sitemap produced during the design phase

- Most sitemaps only list publications and certain pages. Everything else, such as archive pages, PDFs, and the login system has been left out.
- Taxonomy terms, which do not appear on the above design, have, however, been added to the sitemap.
- The sitemap link is often located in the footer. **The page has been renamed site-map (in two words)**, to avoid confusion with its counterpart, the XML index.

Create the site map with Elementor

Here's what the sitemap for **fuyens.ch** looks like.

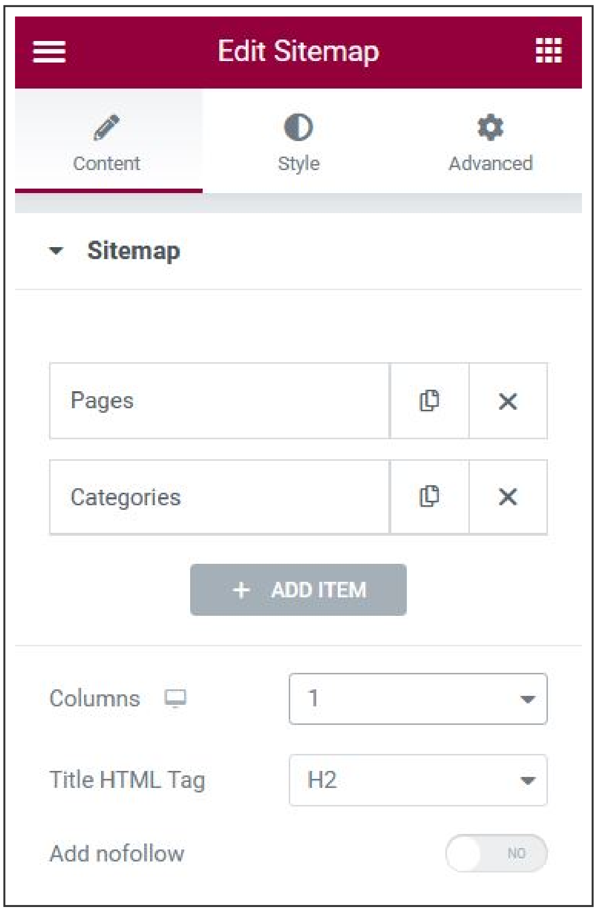
Elementor provides a widget to create the sitemap. If this one is relatively complete, it is quite conceivable to modify it to act on the data display.



The sitemap's widget

To create a sitemap, of course, you need to start by creating a new page, then drag the sitemap widget into a section on it. This will look like the icon list widget, namely, a sequence of items.

It is of course possible to change styles such as color and typography and the number of columns when displayed.


Edit Sitemap

Content | Style | Advanced

Sitemap

Pages

Categories

+ ADD ITEM

Columns: 1

Title HTML Tag: H2

Add nofollow: NO

The sitemap's widget

However, the sitemap is quite basic and does not allow, for example, to create its own title and to change its color (one color per type of publications, for example). As is often the case with Elementor, you have to rely on CSS to achieve this. Here are some tips.

- To delete a title, the class Elementor uses is **"elementor-sitemap-CPT-title"**, where you need to replace **"CPT"** with the type of publications.
- To delete the title of a taxonomy, it's the same thing.
- The CSS property to use is **"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-t
  display: none;}
```

- To handle lists, Elementor uses bullets, but has completely omitted line spacing. Again, CSS comes to the rescue with the **".elementor-sitemap-item a"** class.
- To handle selection on mouseover, we can use the **"hover"** property of the same class as above.

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

- To create the title and publications counter, you need to use the **"Title"** widget.
- To create the counter, you need to use the dynamic parameter **"number of publications"** and set it to display the right type of publications.
- To create the term counter of a taxonomy, unfortunately there is no dynamic parameter available. So you have to use the **"short code"** parameter instead.

Below is a short code example to display the number of terms of the custom type **"it_type"**. The **"hide_empty"** parameter is used to display only terms for which a publication exists.

```
// 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');
```

With a little CSS, it is quite possible to create beautiful sitemaps. So there is no need to use any other plugin than the one provided by Elementor.

The sitemap in XML format

The second sitemap is much more complicated to manage. It involves creating and managing an XML file containing all the pages, taxonomies and other information that the site would like to pass on to search engines. To access a sitemap, you need to enter the URL below:

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

Of course, you have to replace the domain name with your own.

Automatic method with a plugin

The first method to create a sitemap is to use a plugin like**Yoast** or **Rank Math**. All these SEO plugins allow you to make a sitemap more or less automatically. They all have advantages and disadvantages. This guide will not go into comparing the best sitemap plugin, but rather look at alternative methods.

Automatic method without plugin

One of the other methods to create a sitemap is to do it directly with WordPress. In fact, WordPress has built-in this feature since version 5.5.

To enable the automatic sitemap, you need to add this line of PHP in the**“functions.php”** file.

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

Delete Elementor templates

This fully automatic method creates a complete sitemap, too complete. Indeed, with Elementor, template types are considered as pages. You then need to remove them with the code below.

```
// 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');
```

Remove users

In the same regard, it is possible to filter and remove users who have an account on the site. This is the kind of information that has no reason to be indexed by 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);
```

Remove a taxonomy

If we want to remove the taxonomy from the custom type **"IT"**, it's the same code...or almost.

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

All official WordPress information can be found **here**.

Add a provider

What is a provider? According to the documentation of WordPress, a provider is nothing more than a type of content. A page or an article is a type of content. An image is also a separate content type. Custom types, which are articles with other meta-data are also other content types. One type of content that no one thinks of at first are urls. Indeed, urls that do not represent pages or articles, but in the case of Elementor, an archive page or a search result page, are not automatically added to the sitemap. In these cases, a new provider must be created in order to add a new type of content.

Below is an example of a new provider to create custom urls.

```
// 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');
```

Once the **“WP_Custom_Sitemaps”** instance has been created, the **“WP_Sitemaps_Provider”** class must be extended and its own code has to be added. Namely, the base class contains two public functions **“get_url_list”**, which allows to display its custom urls and **“get_max_num_pages”** which allows to display the number of pages. To keep it simple, the example below is based on a single page.

```
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;
  }}
```

Manual approach

The WordPress method above is handy. Despite this, it is also possible to create your own sitemap by hand, starting from a blank sheet of paper. The first reason to do this is certainly the poor handling of multilingualism.

Below are the steps to achieve this.

Rewriting urls

First, you need to rewrite the URL to access your sitemaps(s).

- The first **"rewrite_rule"** will call the sitemap index when entering the url **"https://www.fuyens.ch/sitemap.xml/"**.
- The second **"rewrite_rule"** will call the corresponding sitemap depending on the content type, as well as the page if the number of publications exceeds 2000. For example, the url **"https://www.fuyens.ch/en/sitemap-posts-it-1.xml/"** will display the first page of the sitemap concerning "IT" publications.

```php
<?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');
```

Render the sitemap

The second step is to display the desired sitemap using the **"template_redirect"** functionality.

```
// 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-password', 'ac
  $exclude = exclude_pages ($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');
```

The function that excludes pages is below. It uses Polylang functions to find a publication and its translation. Thus, it is not necessary to list all the publications to be excluded. Only one is needed.

```
// 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;}
```

Render the sitemap index

The third step is about displaying the index. Here are some explanations.

- The first few lines are about setting up an XSL file.
- The next few lines set up the structure of the XML file with its header.
- The **"unset"** group allow unwanted types to be removed. On this example, the "Elementor" templates, the article type and the attachments are not taken into account.
- Then, for each language, each publication type and each taxonomy, the number of pages is calculated and the index links are created using the **"<loc>"** tag.

```
// 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;}
```

The **“get_filtered_posts”** function is described below. It uses the WordPress function **“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' => $numberpost
                                          'paged' => $paged, 'lang' => $lang, 'post_status' => 'publis
  return $posts;}
```

Render the posts of the sitemap

The code looks like the one for the index, except that it is possible to handle the different languages with the **“<xhtml>”** tags and give its translation with the **“alternate”** tag. Of course, you need to install the Polylang plugin, which handles this perfectly.

In addition, it adds the date and time of the last modification of the publication with the**“<lastmod>”** tag.

```
// 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) . '" rel="
          }
        }
      }

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

Render the terms of the sitemap

For each taxonomy, the sitemap will display the terms corresponding to it. The code is identical to the one for 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="al
          }
        }
      }
      $xml .= '</url>';
    }
  }

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

  return $xml;}
```

The style sheet for the sitemap

If HTML has CSS to customize its style, XML also has its style sheet. It carries the extension XSL for **"Extended Stylesheet Language"**. Thanks to it, we can make our stylesheet very pretty. (which in the case of a sitemap file, is not very useful).

The sitemap layout of this site was done by Pedro Borges. You can find the stylesheet on his **github**.

The XSL file downloaded, you need to copy it into your child theme and reference it in the code. The lines of code below will

- Check if your site uses the **"HTTP"** or **"HTTPS"** protocol.
- Get the url of your domain name.
- Get the url of the xsl file, which is located, for this example, in the **"sitemap"** folder of Elementor's **"hello"** child theme.

```
// 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

Creating a sitemap for your website is optional, but highly recommended. First of all, a sitemap in HTML format allows anyone to find a publication or to see the content of a site. You can use the browser search "CTRL+F" on the sitemap page, which is much faster than any search engine. The XML file, instead, allows to accelerate the indexation of the pages at Google, but will not bring any benefit on the ranking. On the other hand, the sitemap is mandatory with most caching plugins such as **"Wp-Rocket"** or **"Wp-Optimize"**. The use of the above makes the sitemap almost indispensable, if you want to display your site's pages quickly.