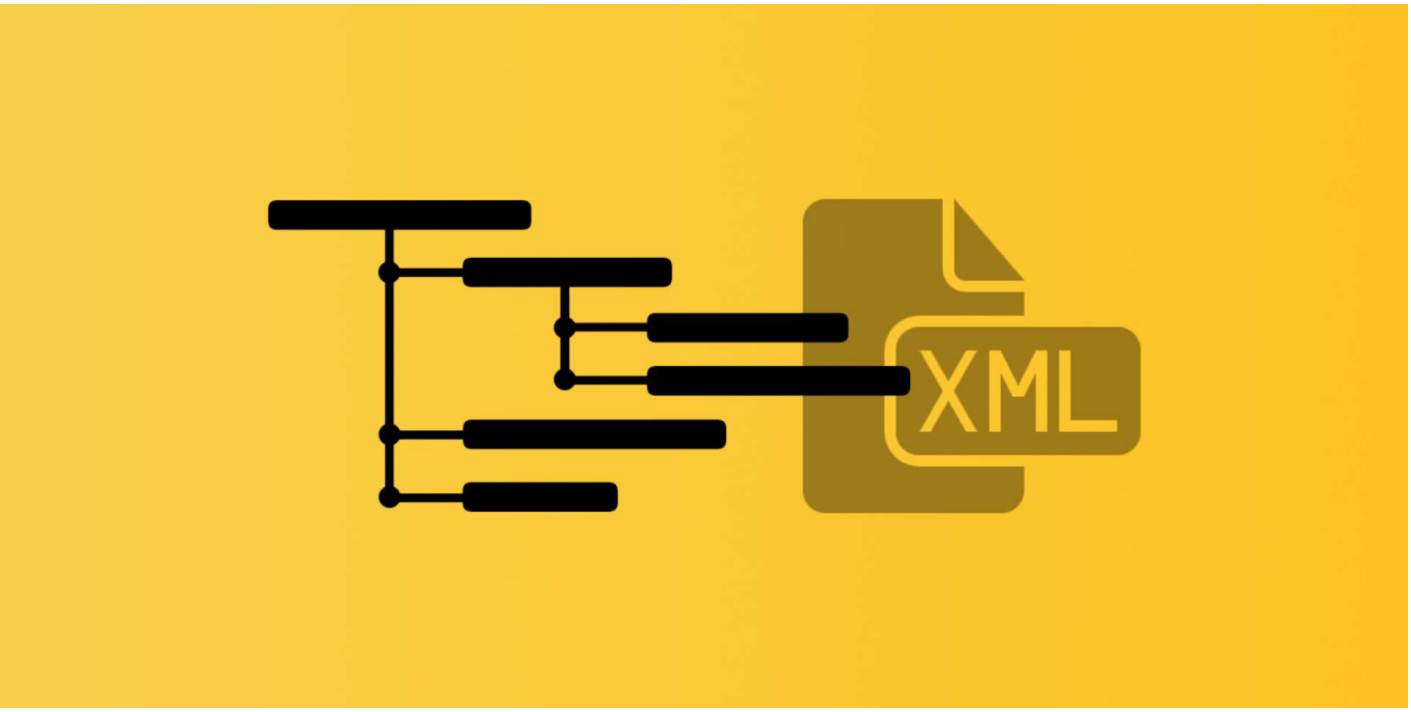
Laravel : Créer un sitemap puis le soumettre à Google



Auteur	Domaine	Technologies
Wilo Ahadi	Développement d'app. web	Laravel, PHP

Un guide pour mettre en place un sitemap (plan de site) dans un projet Laravel pour l'indexation automatique des URLs sur le moteur de recherche Google.

Sommaire

Introduction au sitemap

Sitemap est un protocole qui permet au webmaster d'informer les moteurs de recherches des pages d'un site web disponibles pour l'indexation automatique.

Il s'agit d'un fichier, généralement présenté au format XML ou texte, qui répertorie ou liste les adresses (URL) d'un site web en permettant d'inclure des informations complémentaires pour chaque adresse tels que la date de

dernière modification, la fréquence de mise à jour, la priorité par rapport aux autres adresses, ...

Les moteurs de recherche tels que Google lisent le sitemap pour explorer plus intelligemment un site web, présenter les statistiques sur les recherches qui aboutissent aux pages de ce site, …

Dans ce guide, nous allons voir comment mettre en place un sitemap (plan de site) dynamique au format XML dans un projet Laravel.

Le protocole sitemap

Le protocole sitemap utilise des tags XML et le fichier XML doit être encodé en UTF-8.

Voici l'exemple d'un sitemap au format XML avec une URL renseignée :

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
   <url>
      <loc>http://www.monsite.com/posts</loc>
      <lastmod>2020-06-01</lastmod>
      <changefreq>monthly</changefreq>
      <priority>0.8</priority>
   </url>
</urlset>
```

Décrivons ce schéma :

- `<urlset>` (*requis*) : Encapsule les URLs et indique l'espace de noms (namespace)
- `<url>` (*requis*) : Tag parent pour chaque entré d'URL
- `<loc>` (*requis*) : L'URL de la page
- `<lastmod>` (*optionnel*) : Date de la dernière mise à jour de la page. La valeur doit respecter l'un de formats Date and Time recommandé par W3C.
- `<changefreq>` (*optionnel*) : La fréquence (périodicité) de mise à jour de la page. Les valeurs valides sont « always », « hourly », « daily », « weekly », « monthly », « yearly », « never »
- `<priority>` (*optionnel*) : L'importance de l'URL par rapport aux autres URLs. La valeur doit être comprise entre "0.0" et "1.0". La valeur de priorité par défaut est "0.5".

Seuls les tags `<urlset>`, `<url>` et `<loc>` sont requis. Les tags `<lastmod>` , `<changefreq>` et `<changefreq>` sont au choix.

Notez bien : Dans un fichier XML, la valeur d'un tag (y compris pour une URL) doit utiliser les codes d'échappement pour les caractères énumerés au tableau ci-dessous :

Caractère		Code d'échappement
Ampersand	&	`&amp;`
Single Quote	'	`&apos;`
Double Quote	"	`&quot;`
Greater Than	>	`&gt;`

Less Than	<	<

Mise en place d’un sitemap

Partant du schéma que nous venons de décrire au protocole sitemap, mettons en place un sitemap pour les publications d’un site web que nous supposons être représentées par le modèle « Post » (table « posts ») dans un projet Laravel.

Notez Bien : Pour bien organiser l’application, nous vous recommandons de toujours créer un sitemap spécifique à chaque modèle, c’est-à-dire aussi une route spécifique pour chaque sitemap comme nous allons le faire en trois étapes :

1. La route

Au fichier **routes/web.php**, ajoutons la ligne suivante :

```
Route::get("sitemap/posts", "SitemapController@posts")->name("sitemap.posts");
```

Cette URL « sitemap/posts » (GET) nommée « sitemap.posts » est gérée par la méthode « posts » du contrôleur « SitemapController ».

Si vous le désirez, vous pouvez ajouter l’extension à l’URL : « sitemap/posts.xml »

2. Le contrôleur

Générons le contrôleur « SitemapController » en exécutant la commande *artisan* suivante :

```
php artisan make:controller SitemapController
```

Au fichier **App/http/SitemapController.php** que nous avons maintenant, décrivons la méthode « posts » où nous récupérons les publications (données) « Post » de la base de données puis retournons la réponse XML avec la vue « sitemaps.posts » et les données :

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Post;

class SitemapController extends Controller
{
    public function posts () {
     $posts = Post::latest()->get();
     return response()->view('sitemaps.posts', compact('posts'))->header('Content-Ty
    }
}
```

3. La vue

Créons le fichier **ressources/views/sitemaps/posts.blade.php** où nous parcourons les différentes publications (collection de « Post ») en créant pour chacune d'elles une `<url>` du sitemap :

```
@php echo '<?xml version="1.0" encoding="UTF-8"?>' @endphp
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
 @foreach ($posts as $post)
 <url>
   <loc>{{ route('posts.show', $post) }}</loc>
   <lastmod>{{ $post->updated_at->tz('UTC')->toAtomString() }}</lastmod>
   <changefreq>monthly</changefreq>
   <priority>0.8</priority>
 </url>
 @endforeach
</urlset>
```

Notez la façon dont nous affichons à la première ligne le code `<?xml version …>` en PHP, c'est pour qu'il n'y ait pas de confusion entre XML et PHP.

L'index des sitemaps

Nous vous avions recommandé de créer un sitemap spécifique à chaque modèle pour bien organiser l'application, ce qui peut vous faire plusieurs sitemaps si vous avez plusieurs modèles.

Il faut aussi noter qu'un fichier sitemap ne peut contenir plus de 50000 URLs, ni peser plus de 50Mo (52,428,800 bytes) d'où il faut encore avoir plusieurs sitemaps si on va loin de ces limites.

Si vous avez plusieurs sitemaps, vous pouvez les grouper (les lister) dans un fichier **sitemap index**. Il ne doit lui aussi avoir plus de 50000 sitemaps, ni peser plus de 50Mo.

Voici l'exemple d'un sitemap index qui liste deux sitemaps :

```
<?xml version="1.0" encoding="UTF-8"?>
<sitemapindex xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
   <sitemap>
      <loc>http://www.example.com/sitemap1.xml</loc>
      <lastmod>2020-03-02T22:31:39+00:00</lastmod>
   </sitemap>
   <sitemap>
      <loc>http://www.example.com/sitemap2.xml</loc>
      <lastmod>2020-04-01</lastmod>
   </sitemap>
</sitemapindex>
```

Décrivons ce schéma :

- `<sitemapindex>` (*requis*) : Encapsule les informations de tous les sitemaps et indique le namespace
- `<sitemap>` (*requis*) : Tag parent pour chaque entrée de sitemap

- `<loc>` (*requis*) : L'URL du sitemap
- `<lasmod>` (*optionnel*) : La date de la dernière mise à jour du sitemap. Sa valeur doit respecter l'un de formats Date and Time recommandé par W3C

Complétons le sitemap que nous avons mis en place précédemment en ajoutant pour le sitemap index :

- La route « sitemap/index »
- La méthode « index » au contrôleur « SitemapController »
- La vue « index.blade.php »

1. La route

Ajoutons au fichier **routes/web.php** la route GET « sitemap/index » qui va être géré par la méthode « index » du contrôleur « SitemapController ». Nommons-la « sitemap.index » :

```
Route::get("sitemap/index", "SitemapController@index")->name("sitemap.index");
```

2. Le contrôleur

La méthode « index » du contrôleur « SitemapController » récupère la dernière entrée « Post » dont on utilisera la date pour indiquer la dernière mis à jour du sitemap « sitemap/posts », puis retourne la reponse XML avec la vue « sitemaps.index » et les données :

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Post;

class SitemapController extends Controller
{
 // Sitemap index
 public function index () {
  $post = Post::latest()->first();
  return response()->view("sitemaps.index", compact('post'))->header('Content-Type',
 }

 // Sitemap posts ...
}
```

3. La vue

Nous allons supposer avoir un sitemap supplémentaire à l'URL GET « sitemap/pages » qui liste les pages du site web qui ne dépendent pas d'un modèle; Question d'avoir au moins deux sitemaps au sitemap index.

Code source de la vue **ressources/views/sitemaps/pages.blade.php** :

```
@php echo '<?xml version="1.0" encoding="UTF-8"?>' @endphp
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
```

```
  <url>
    <loc>{{ route('welcome') }}</loc>
  </url>
  <url>
    <loc>{{ route('login') }}</loc>
  </url>
  <url>
    <loc>{{ route('register') }}</loc>
  </url>
</urlset>
```

La vue **ressources/views/sitemaps/index.blade.php** se présente donc de la manière suivante :

```
@php echo '<?xml version="1.0" encoding="UTF-8"?>' @endphp
<sitemapindex xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
   <sitemap>
      <loc>{{ route('sitemap.pages') }}</loc>
   </sitemap>
   <sitemap>
      <loc>{{ route('sitemap.posts') }}</loc>
      <lastmod>{{ $post->updated_at->format("Y-m-d") }}</lastmod>
   </sitemap>
</sitemapindex>
```

Nous allons utiliser l'adresse du sitemap index au point suivant pour soumettre tous les sitemaps du site web aux moteurs de recherches.

Soumettre un sitemap à Google

Maintenant que nous avons l'index de tous les sitemaps de notre site web, voyons comment l'envoyer au moteur de recherche Google pour l'indexation automatique des URLs.

Google Search Console (GSC)

Google Search Console est une service gratuit de Google qui permet aux webmasters de contrôler et maintenir la présence de leur site web sur le moteur de recherche Google

Il met à notre disposition des outils et des rapports qui permettent de :

- Valider un nom de domaine pour en soumettre un sitemap
- Consulter les statistiques de trafic de la recherche Google d'un site web
- Demander l'indexation d'un nouveau contenu ou la reindexation d'un contenu mis à jour
- Recevoir les alertes et résoudre les problèmes d'indexation
- Découvrir les sites web qui référencent le votre
- ....

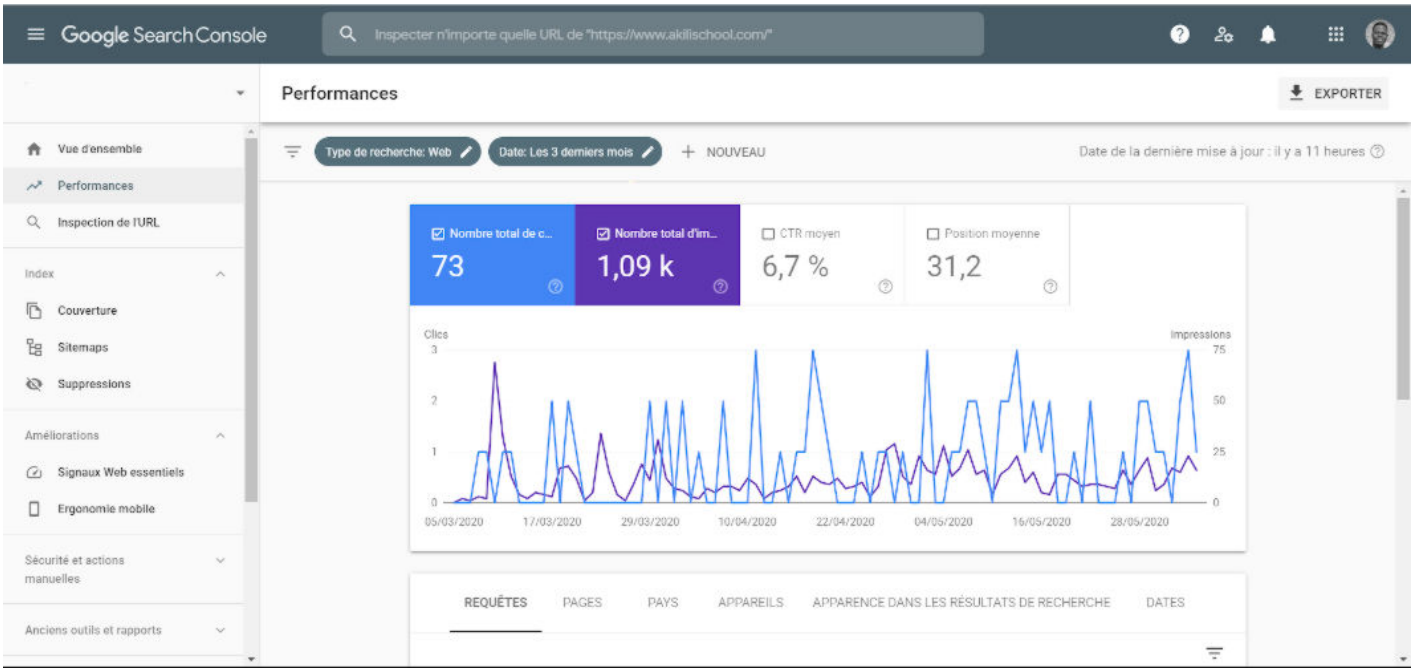
Voici la procédure pour envoyer un sitemap index à Google :

1. Accédez à la plateforme Google Search Console , allez sur **Commencer maintenant** puis connectez-

vous avec votre compte *@gmail.com*

2. Sur la page d'accueil GSC, il est demandé d'ajouter une propriété pour laquelle vous souhaitez soumettre un sitemap. Entrez soit un domaine ou un préfixe de l'URL puis cliquez sur **Continuer**. La procédure de validation vous sera indiquée.
3. Une fois le nom de domaine validé, sélectionnez-le au menu à gauche dans **Propriété de recherche**
4. Toujours au menu à Gauche de votre propriété, allez sur **Sitemaps**
5. Entrez l'adresse de votre sitemap index (Ex: http://monsite.com/sitemap/index) au champ "Ajouter un sitemap"
6. Cliquez sur **Valider**

Les robots d'exploration Google vont progressivement explorer votre sitemap, parcourir les URLs, ... puis commencer à vous présenter l'évolution de votre site web sur le moteur de recherche :



Packages de sitemap pour Laravel

En parcourant le web, je suis tombé sur quelques packages qui traitent le protocole sitemap, voici deux d'entre eux qui peuvent vous inspirer :

- spatie/laravel-sitemap
- Laravelium/laravel-sitemap