

Beste de savoir

Utiliser le DOM pour générer un
sommaire en PHP

28 mai 2021

Table des matières

1. Extraire le DOM	1
2. Parcourir le DOM	3
3. Construire le HTML du sommaire	4

Imaginez : vous avez créé un site vous permettant d'écrire des pages à la volée, par exemple via un parseur Markdown (ou BBCode si vous avez de la bouteille) ou via un éditeur **WYSIWYG**, mais vous voulez générer automatiquement des sommaires pour faciliter la navigation de vos utilisateurs.

Vous avez alors deux solutions:

1. *Utiliser du JavaScript côté client*, imposant un délai et nécessitant des ressources en plus pour l'utilisateur
2. *Utiliser PHP côté serveur* pour servir la page avec le sommaire directement, y compris aux robots de référencement

Pragmatiques, vous préférez un site léger pour l'utilisateur : **c'est parti pour le faire en PHP !**



Problème

Vous avez besoin de traiter un **DOM** partiel et plusieurs niveaux de titres, en plus de vouloir afficher le sommaire où vous le souhaitez.

Voyons donc comment utiliser le **DOM** en PHP pour faire tout ça nous-même, de A à Z!

1. Extraire le DOM

Partons du principe que l'on a une chaîne de caractère qui contient notre HTML (par exemple obtenu à partir d'un convertisseur de Markdown, ou depuis un éditeur **WYSIWYG**).

Pour faire simple je vais définir un fichier HTML...

```
1 <p>Un texte d'intro</p>
2
3 <h1>Un premier titre</h1>
4
5 <h2>Un sous-titre</h2>
6 <p>Du texte</p>
```

1. Extraire le *DOM*

```
7
8 <h2>Un autre sous-titre</h2>
9 <h3>Le PHP c'est génial</h3>
10 <p>Encore du texte</p>
11 <h3>On peut faire plein de choses !</h3>
12 <p>Toujours du texte</p>
13
14 <h1>Un second titre</h1>
15
16 <h2>Encore un sous-titre</h2>
17 <h3>Le PHP c'est vraiment génial</h3>
18 <p>Encore plus de texte</p>
19 <h3>On peut faire tout plein de choses !</h3>
20 <p>Toujours plus de texte</p>
21
22 <h2>Encore un autre sous-titre</h2>
23 <p>Vous reprendrez bien un peu de texte ?</p>
```



Attention

Ce tutoriel n'a pas pour but de vous apprendre la sécurité, n'oubliez pas de vous protéger contre les failles (notamment *XSS*) avant de traiter ou servir du contenu venant de l'extérieur.

... et le lire avec PHP :

```
1 <?php
2 $page_html = file_get_contents('texte.html');
3 $page_html = mb_convert_encoding($page_html, 'HTML-ENTITIES',
   'UTF-8'); // On s'assure d'avoir de l'UTF-8
```



Attention

Votre HTML doit être valide pour pouvoir le parser, sans quoi vous pourriez rencontrer des erreurs par la suite

On commence alors par parser ce texte pour obtenir un *DOM*. Heureusement PHP a tout ce qu'il faut pour faire ce travail à notre place :

```
1 $page_dom = new DOMDocument('2.0', 'UTF-8'); // Mon texte HTML
   étant encodé en UTF-8, je peux préciser cet encodage au parseur
2 $page_dom->loadHTML($page_html, LIBXML_HTML_NOIMPLIED |
   LIBXML_HTML_NODEFDTD); // On précise au parseur de ne pas
   ajouter de `doctype` ou de balises `<html><body>` inutiles
```

2. Parcourir le *DOM*

On obtient alors un objet représentant un *DOM* que l'on pourra alors parcourir pour récupérer les infos qui nous intéressent.

2. Parcourir le *DOM*

On peut alors créer une fonction qui, à partir d'un *DOM*, récupère les titres jusqu'au niveau désiré.

```
1 <?php
2 if (!function_exists('extract_toc')) {
3     function extract_toc(DOMDocument $dom, int $max_level = 6) {
4         // Notre code viendra s'insérer ici
5     }
6 }
```

Avant de pouvoir parcourir notre *DOM*, il faut définir un chemin à suivre. Pour cela on utilise XPath, comme en XML :

```
1 $xpath = new DOMXPath($dom);
2 $xpath->registerNamespace('html', 'http://www.w3.org/1999/xhtml');
```

On peut ensuite calculer la liste des titres à récupérer :

```
1 $max_level = min(max($max_level, 1), 6); // Les titres en HTML vont
   de h1 à h6 maximum
2
3 $headings_queries = [];
4
5 foreach (range(1, $max_level) as $h) {
6     $headings_queries[] = 'self::h'.$h; // Les requêtes XPath
   partent du nœud courant, pour chercher les balises `hX`
7 }
```

i

À noter que l'on pourrait raccourcir ce code en utilisant `array_map` par exemple, mais je préfère garder un code simple pour le moment. 🍊

On peut ensuite construire une requête unique qui nous donnera l'ensemble des titres d'un coup :

3. Construire le HTML du sommaire

```
1 $query_headings = implode(' or ', $headings_queries); // On
   sélectionne toutes les balises `h1` ou `h2` ou `h3`, etc.
2 $query = '//*[@'.$query_headings.']'; // On part de la racine pour
   chercher les balises en question
3 $headings = $xpath->query($query);
```

3. Construire le HTML du sommaire

Maintenant que l'on a récupéré les titres, il va falloir construire notre sommaire !

Commençons par créer une liste en HTML :

```
1 if (count($headings)) {
2     $toc = '<ol class="toc-level-1">';
3
4     // La suite du code va venir ici, pour construire la liste
5
6     $toc .= '</ol>';
7 }
8
9 return $toc ?? '';
```

Si on veut imbriquer les différents niveaux, on peut initialiser quelques variables :

```
1 $current_level = 1;
2 $items = 0;
```

i

J'utilise les fonctions `url_title` et `xss_clean` ci-dessous qui ne sont pas natives à PHP, il faudra donc les définir vous-même pour sécuriser votre application.

Vous pouvez recréer ces fonctions assez simplement:

- `url_title` permet ici de transformer un texte en *kebab-case*
- `xss_clean` filtre le texte pour empêcher les failles XSS (de façon un peu plus complète que `htmlspecialchars`) avant de l'envoyer aux clients

C'est parti, bouclons sur nos titres pour construire notre sommaire !

```
1 foreach ($headings as $n_i => $node){
2     $level = (int) $node->tagName{1};
3     $node_id = $node->getAttribute('id');
```

3. Construire le HTML du sommaire

```
4
5     if (empty($node_id)) {
6         // On profite de l'occasion pour ajouter un ID à notre
7         titre directement dans le DOM, pour pouvoir utiliser des ancrés
8         $node_id = 'toc_'.url_title(strip_tags($node->textContent),
9         '-', TRUE);
10        $node->setAttribute('id', $node_id);
11    }
12
13    // On crée un lien vers le titre en question
14    $new_toc = '<a
15    href="#"'. $node_id. '">'.xss_clean($node->textContent). '</a>';
16
17    // =====
18    // On va créer les éléments de la liste ici...
19    //          Le code est un peu plus bas
20    // =====
21
22    $current_level = $level;
23    $items++;
24 }
```



La fonction `strip_tags` [↗](#) permet de supprimer les balises HTML contenues dans chaque titre pour ne pas polluer cet attribut `id`

Et maintenant pour la création du contenu de la liste, en gérant l'imbrication :

```
1  if ($level > $current_level) {
2      // On monte d'un ou plusieurs niveaux, on crée une nouvelle
3      liste
4      for ($a = 0; $a < $level-$current_level; $a++) {
5          $toc .= '<ol class="toc-level-' . $level. '"><li>';
6      }
7      $toc .= $new_toc;
8      $items = 1;
9  }
10 elseif ($level === $current_level) {
11     $toc .= ($items ? '</li>' : ''). '<li>'. $new_toc;
12     $items++;
13 }
14 else {
15     // On descend d'un niveau, on ferme la liste
16     for ($a = 0; $a < $current_level-$level; $a++) {
17         $toc .= '</li></ol>';
18     }
19     $toc .= '</li><li>'. $new_toc;
20 }
```

3. Construire le HTML du sommaire

```
19     $items = 0;  
20 }
```

i

Notez que l'on ne ferme pas les `` juste après les liens, cela permet de gérer l'imbrication des `` au bon endroit.

?

Mais... les listes ne sont pas bien fermées !

C'est vrai ! Il faut donc, après la boucle, fermer les listes ouvertes :

```
1 for ($a = $level - 1; $a >= 0; $a--) {  
2     $toc .= '</li></ol>';  
3 }
```

On peut maintenant utiliser notre fonction, en lui fournissant le **DOM** à parcourir, pour récupérer le nouveau HTML (celui avec les ancres) :

```
1 $page_toc = extract_toc($page_dom);  
2  
3 $page_html = $page_dom->saveHTML(); // On peut sauvegarder le DOM  
   mis à jour par la fonction `extract_toc` pour bénéficier des  
   ancres  
4  
5 echo $page_toc; // On affiche notre sommaire  
6  
7 echo $page_html; // On affiche le contenu de notre HTML
```

Vous pouvez comparer votre code au mien [sur GitHub](#) si vous le souhaitez.

?

Bonus

Pour les plus motivés d'entre vous, vous pouvez utiliser une *fonction récursive* pour générer le sommaire... qui s'en sent capable ?

Encore mieux, vous pouvez aussi *créer un DOM* pour créer ces listes... vous êtes prêts ?

Icône du tutoriel par *Lil Squid* via *Noun Project*

Liste des abréviations

DOM Document Object Model. 1–3, 6

WYSIWYG What You See Is What You Get. 1