How to generate a Sitemap.xml file for Bookstack (Working!)

BookStack does not have a built-in function to generate an XML sitemap (which can be submitted to search engines).

However, the BookStack API Scripts GitHub repository contains a PHP script that can be used to generate an XML sitemap for a BookStack instance.

Procedure

1. Download the PHP script into the BookStack directory to the default directory (cd ~)
2. Create an API token in Bookstack (Click on the user on the top right and then under **My Account** > **Access and Security**)
3. Create a Bash script named `generate-sitemap.sh`
4. make it executable: chmod +x generate-sitemap.sh
5. Run the script

```
#!/bin/bash

# Put this file in your default directory (cd /~)
# Change your_website_url with your website url, like https://ffuitleggen.nl
export BS_URL=https://your_wesite_url

# Change ApiToken with the one you generated in Bookstack
export BS_TOKEN_ID=ApiToken

# Change ApiTokenSecret with the one you generated in Bookstack
export BS_TOKEN_SECRET=ApiTokenSecret

# Running the script
php generate-sitemap.php
```

```
# Move Sitemap into public dir
# Change /var/www/bookstack with the root directory of your bookstack if needed
mv sitemap.xml /var/www/bookstack
```

In the script, you define the URL of your BookStack instance. This URL is written into the sitemap. Additionally, you provide the API token and its secret. All three values are set as environment variables and read by the PHP script.

Since the PHP script always generates the sitemap file in the same directory, I move it to the `public` directory at the end.

The script

This script will scan through all pages, chapters books and shelves via the API to generate a sitemap XML file. **This is a very simplistic single-script-file example of using the endpoints API together** , it is not a fully-featured & validated script.

https://github.com/BookStackApp/api-scripts/blob/main/php-generate-sitemap/generate-sitemap.php

Keep in mind, The sitemap generated will reflect content visible to the API user used when running the script.

`generate-sitemap.php`

```
#!/usr/bin/env php
<?php

// API Credentials
// You can either provide them as environment variables
// or hard-code them in the empty strings below.
$baseUrl = getenv('BS_URL') ?: '';
$clientId = getenv('BS_TOKEN_ID') ?: '';
$clientSecret = getenv('BS_TOKEN_SECRET') ?: '';

// Output File
// Can be provided as a arguments when calling the script
// or be hard-coded as strings below.
$outputFile = $argv[1] ?? './sitemap.xml';
```

```
// Script logic
////////////////

// Check we have required options
if (empty($outputFile)) {
    errorOut("An output file needs to be provided");
}

// Create the output folder if it does not exist
$outDir = dirname($outputFile);
if (!is_dir($outDir)) {
    mkdir($outDir, 0777, true);
}

// Clean up the base path
$baseUrl = rtrim($baseUrl, '/');

// Additional endpoints not fetched via API entities
$nowDate = date_format(new DateTime(), 'Y-m-d');
$additionalEndpoints = [
    ['endpoint' => '/', 'updated' => $nowDate],
    ['endpoint' => '/books', 'updated' => $nowDate],
    ['endpoint' => '/search', 'updated' => $nowDate],
    ['endpoint' => '/login', 'updated' => $nowDate],
];

// Get all shelf URLs
$shelves = getAllOfAtListEndpoint("api/shelves", []);
$shelfEndpoints = array_map(function ($shelf) {
    return ['endpoint' => '/shelves/' . $shelf['slug'], 'updated' => $shelf['updated_at']];
}, $shelves);

// Get all book URLs and map for chapters & pages
$books = getAllOfAtListEndpoint("api/books", []);
$bookSlugsById = [];
$bookEndpoints = array_map(function ($book) use (&$bookSlugsById) {
    $bookSlugsById[$book['id']] = $book['slug'];
    return ['endpoint' => '/books/' . $book['slug'], 'updated' => $book['updated_at']];
}, $books);
```

```
// Get all chapter URLs and map for pages
$chapters = getAllOfAtListEndpoint("api/chapters", []);
$chapterEndpoints = array_map(function ($chapter) use ($bookSlugsById) {
    $bookSlug = $bookSlugsById[$chapter['book_id']];
    return ['endpoint' => '/books/' . $bookSlug . '/chapter/' . $chapter['slug'], 'updated' =>
$chapter['updated_at']];
}, $chapters);

// Get all page URLs
$pages = getAllOfAtListEndpoint("api/pages", []);
$pageEndpoints = array_map(function ($page) use ($bookSlugsById) {
    $bookSlug = $bookSlugsById[$page['book_id']];
    return ['endpoint' => '/books/' . $bookSlug . '/page/' . $page['slug'], 'updated' => $page['updated_at']];
}, $pages);

// Gather all our endpoints
$allEndpoints = array_merge(
    $additionalEndpoints,
    $pageEndpoints,
    $chapterEndpoints,
    $bookEndpoints,
    $shelfEndpoints
);

// Fetch our sitemap XML
$xmlSitemap = generateSitemapXml($allEndpoints);
// Write to the output file
file_put_contents($outputFile, $xmlSitemap);

/**
 * Generate out the XML content for a sitemap
 * for the given URLs.
 */
function generateSitemapXml(array $endpoints): string
{
    global $baseUrl;
    $doc = new DOMDocument("1.0", "UTF-8");
    $doc->formatOutput = true;
    $urlset = $doc->createElement('urlset');
    $urlset->setAttribute('xmlns', 'http://www.sitemaps.org/schemas/sitemap/0.9');
```

```
    $doc->appendChild($urlset);
    foreach ($endpoints as $endpointInfo) {
        $date = (new DateTime($endpointInfo['updated']))->format('Y-m-d');
        $url = $doc->createElement('url');
        $loc = $url->appendChild($doc->createElement('loc'));
        $urlText = $doc->createTextNode($baseUrl . $endpointInfo['endpoint']);
        $loc->appendChild($urlText);
        $url->appendChild($doc->createElement('lastmod', $date));
        $url->appendChild($doc->createElement('changefreq', 'monthly'));
        $url->appendChild($doc->createElement('priority', '0.8'));
        $urlset->appendChild($url);
    }

    return $doc->saveXML();
}

/**
 * Consume all items from the given API listing endpoint.
 */
function getAllOfAtListEndpoint(string $endpoint, array $params): array
{
    $count = 100;
    $offset = 0;
    $all = [];

    do {
        $endpoint = $endpoint . '?' . http_build_query(array_merge($params, ['count' => $count, 'offset' =>
$offset]));
        $resp = apiGetJson($endpoint);

        $total = $resp['total'] ?? 0;
        $new = $resp['data'] ?? [];
        array_push($all, ...$new);
        $offset += $count;
    } while ($offset < $total);

    return $all;
}
```

```
/**
 * Make a simple GET HTTP request to the API.
 */
function apiGet(string $endpoint): string
{
    global $baseUrl, $clientId, $clientSecret;
    $url = rtrim($baseUrl, '/') . '/' . ltrim($endpoint, '/');
    $opts = ['http' => ['header' => "Authorization: Token {$clientId}:{$clientSecret}"]];
    $context = stream_context_create($opts);
    return @file_get_contents($url, false, $context);
}

/**
 * Make a simple GET HTTP request to the API &
 * decode the JSON response to an array.
 */
function apiGetJson(string $endpoint): array
{
    $data = apiGet($endpoint);
    return json_decode($data, true);
}

/**
 * DEBUG: Dump out the given variables and exit.
 */
function dd(...$args)
{
    foreach ($args as $arg) {
        var_dump($arg);
    }
    exit(1);
}

/**
 * Alert of an error then exit the script.
 */
function errorOut(string $text)
{
    echo "ERROR: " . $text;
    exit(1);
```

```
}
```

Revisie #1
Gemaakt: 30 mei 2025 07:36:59 door Gert
Bijgewerkt: 30 mei 2025 07:37:32 door Gert