Managing Sitemap XML with Google Search Console

With this next installment of our series on Google Search Console, we're going to cover how Google reports your site's indexing. Simply put, indexing describes which pages Google stores from your site – it's the database from which the search engine draws when it's assembling search results pages (SERPs). (If you missed it, the first installment of this series is on getting started with Google Search Console.)

The count of these stored pages, labeled "Valid" in the screenshot below, indicates how many of your pages Google has ready for search appearance. Google keeps information about more of your site's pages than those marked Valid, and not all pages have equal value. A lively homepage is vastly different than a static privacy policy, for example.

This helps you know where your site stands, with regard to indexing, and you can also take advantage of factors that influence the indexing of your site, primarily Google’s Sitemap XML protocol. Search Console allows you to manage this indexing tool for your site.

Index Coverage

The most important information to look at on this report is Index Coverage report errors, highlighted in red. Ideally, you want to see zero errors, but when crawl errors do occur Google cannot index your pages. These errors populate the bar graph and are listed in the data table below with details for troubleshooting.

The information panels at the top of the bar graph act as toggles just like the Performance Report graph and Index Coverage has similar message indicators with pop-up information panels as well.

Message opens an information panel.

You might find the Index Coverage bar chart is easier to analyze information, even with all toggles switched on, as compared to the Performance Report chart. Toggle the Impressions checkbox to show a line graph with Impressions volume on a secondary axis.

Keep in mind when navigating away and back again, the state of your toggles reverts back to the default. You will have to switch toggles back on again.

Index Coverage Full Bar Chart

The detail table provides a set of filters that is somewhat like the Performance Report data table, but you might find it less useful. Each finding is nicely grouped and organized into rows. Keep these filters in mind, however, when you are working with long time spans and large numbers of pages.

Nicely organized by default

When a detail row has only one page, clicking it will take you to the URL Inspection Tool for that page. Otherwise, clicking will expand to an Index Coverage report of the data selected in that row. Excluded pages should make sense for your site. For example, sites that provide control to users who actively rename and delete things will naturally collect a URL history in their wake.

For example, artists may change the name of their online gallery from time to time or delete artwork that has sold from their websites. You’re going to want a process for keeping that URL history so that when Google crawls the old gallery

page it is redirected, or when it looks for artwork that’s been sold, it gets Status 404. It’s good indexing hygiene to actively work your URL history with Search Console reports and your Sitemap XML.

More about the Google Search Console and SEO for developers

Sitemap XML

There is some “inclusion” history regarding XML for search engines. It’s nice that webmasters can still describe their site pages in a format that includes metadata. Google decentralized Paid Inclusion indexes to open them up freely to the webmaster community at large, which entails publishing XML at locations specified in your robots.txt file.

Most commonly, you will need only one Sitemap XML file located at your root directory and spelled ‘sitemap.xml,’ as long as it describes fewer than 50,000 pages. Google’s specification does not allow one sitemap file to describe more. You will need to supply additional sitemaps for coverage of sites that are larger than 50,000 pages if you want to describe that many.

Click a Sitemap row to open Index Coverage for that selected data

Index Coverage Report refined to the Sitemap’s URLs

Authoring Sitemap XML

Static sitemaps can get you going in a pinch, but you really want to use the same framework language used by the site in order to build a dynamic Sitemap file in response to requests for the /sitemap.xml URL. If you’re going with a NodeJS framework, then the following Rails implementation can serve as an example for ideas to build it in React, Angular, or Vue.

First, define a route for the Sitemap and point it to a controller action. The Sitemap controller can then be set to respond with XML using the ‘respond_to :xml’ helper method and

symbol argument. Inside the ‘index’ action definition block, you define instance variables for the database entities matching your criteria, and sort.

Rails Sitemap Controller

Next, use your interpolated (Ruby in this case) view template to craft logic for developing your XML nodes, beginning with the homepage on down. Here is where you get to define metadata such as Priority, and provide last-modified details etc. The homepage can be set at Priority value 1.0, whereas, all other pages should be a fraction thereof, down to lesser pages valued at .3 or lower.

Sitemap definition and Homepage Node

Avoid thinking of Priority in terms of some real-world page value. Instead, think of Priority as how you would like to prioritize the refresh cycle queue based in tandem with anticipated change frequency. One can assume that category and subcategory pages are important, and these should be given higher priority if the pages change frequently.

Building Sitemap Page Nodes

The Sitemap XML page is the one page where response time isn’t going to negatively affect your rankings. Google will be patient enough within reason to successfully fetch metadata up to the 50,000-page mark. It’s already making some assumptions but you definitely don’t want a Sitemap page request to hang. A few thousand pages shouldn’t concern you much. When you have close to 50,000, then make sure it still works efficiently.

When a site’s pages change a lot, you want a Sitemap builder in your site code. You may or may not have luck configuring server-parsed directives to include the execution of a CGI program for this, or point to a CGI program URL as the Sitemap, but that can seem contrived when you can simply program a periodic Sitemap file as program output. You lose the dynamic nature of server-parsed when it’s periodic, but it’s definitely an adequate approach.

In a pinch, static Sitemaps can be fairly safe to build on production hosts using the vi(m) command line editor. Vi is virtually guaranteed to be available for those times when you are afforded a terminal shell. Edit a list of URLs by programming recordings and then playback to generate XML nodes

down your list.

Then you can start working in earnest with Google Search Console, even with just a static Sitemap. Knowing vi(m) is an advantage for Tech SEO practitioners because of exactly this kind of work and you'll want to write something that keeps it up to date.

About The Author

Detlef Johnson is Editor at Large for Third Door Media. He writes a column for Search Engine Land entitled "Technical SEO for Developers." Detlef is one of the original group of pioneering webmasters who established the professional SEO field more than 20 years ago. Since then he has worked for major search engine technology providers, managed programming and marketing teams for Chicago Tribune, and consulted for numerous entities including Fortune 500 companies. Detlef has a strong understanding of Technical SEO and a passion for Web programming. As a noted technology moderator at our SMX conference series, Detlef will continue to promote SEO excellence combined with marketing-programmer features and webmaster tips.
http://feeds.searchengineland.com/~r/searchengineland/~3/stN9woWfzKU/managing-sitemap-xml-with-google-search-console-311790