



SEARCH ENGINE OPTIMIZATIONS ESSENTIALS

**BEST PRACTICES & TECHNIQUES
FOR SEO OPTIMIZATION**

Date - Friday, September 20, 2024

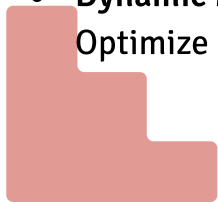
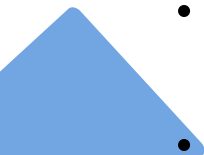
Press Space for next page →



Next.js SEO Guide

Boost your Next.js site's search engine visibility with these essential techniques:

- **Site Map XML**
Generate a dynamic sitemap to help search engines index your content efficiently.
- **Robot.txt**
Control which pages and sections of your website are accessible to crawlers.
- **Static Metadata**
Define consistent SEO metadata for static pages to improve ranking and sharing.
- **Individual Metadata**
Customize metadata for specific pages or sections for targeted optimization.
- **Dynamic Metadata**
Optimize pages with dynamic content by automatically generating metadata for each entry.



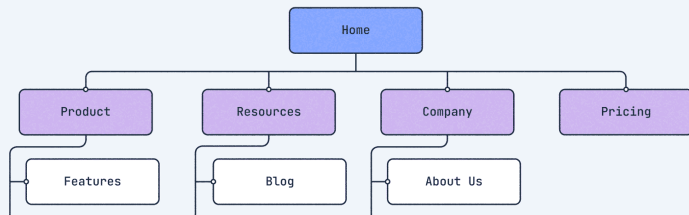
🌐 Setting Up a Sitemap in Next.js

Sitemaps are the easiest way to communicate with Google. They indicate the URLs that belong to your website and when they update so that Google can easily detect new content and crawl your website more efficiently. A sitemap helps search engines efficiently crawl your site's URLs.

This guide explains how to set up a sitemap in **Next.js** using:

- **Pages Router**
- **App Router**

Sitemap example



PAGES ROUTER

THERE ARE TWO OPTIONS:

-Manual

```

<!-- public/sitemap.xml -->
<xml version="1.0" encoding="UTF-8">
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  <url>
    <loc>http://www.example.com/foo</loc>
    <lastmod>2021-06-01</lastmod>
  </url>
</urlset>
</xml>

```

Learn more about [sitemap.xml](#) in the official Next.js documentation.

-getServerSideProps

```
//pages/sitemap.xml.js
const EXTERNAL_DATA_URL = 'https://jsonplaceholder.typicode.com/posts'

function generateSiteMap(posts) {
  return `<?xml version="1.0" encoding="UTF-8"?>
    <urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
      <!--We manually set the two URLs we know already-->
      <url>
        <loc>https://jsonplaceholder.typicode.com</loc>
      </url>
      <url>
        <loc>https://jsonplaceholder.typicode.com/guide</loc>
      </url>
      ${posts
        .map(({ id }) => {
          return `
<url>
<loc>${`${EXTERNAL_DATA_URL}/${id}`}</loc>
</url>
`
        })
        .join('')}
    </urlset>
  `
}
```

App Router

THERE ARE TWO OPTIONS:

Manual

```
//pages/sitemap.ts
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  <url>
    <loc>https://acme.com</loc>
    <lastmod>2023-04-06T15:02:24.021Z</lastmod>
    <changefreq>yearly</changefreq>
    <priority>1</priority>
  </url>
  <url>
    <loc>https://acme.com/about</loc>
    <lastmod>2023-04-06T15:02:24.021Z</lastmod>
    <changefreq>monthly</changefreq>
    <priority>0.8</priority>
  </url>
  <url>
    <loc>https://acme.com/blog</loc>
    <lastmod>2023-04-06T15:02:24.021Z</lastmod>
    <changefreq>weekly</changefreq>
```

Generating a sitemap using code (.js, .ts)

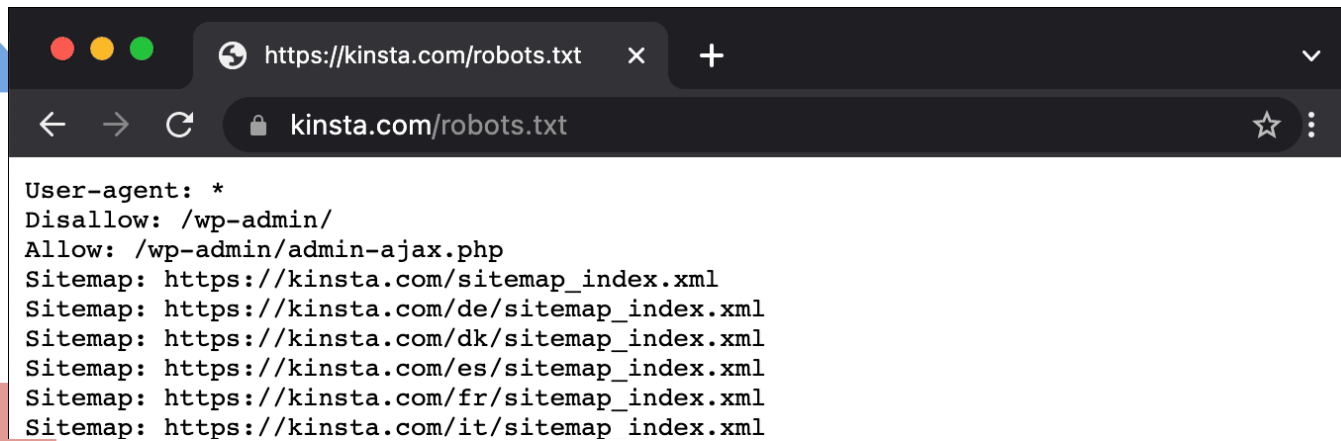
```
//pages/sitemap.ts
import type { MetadataRoute } from 'next'

export default function sitemap(): MetadataRoute.Sitemap {
  return [
    {
      url: 'https://acme.com',
      lastModified: new Date(),
      changeFrequency: 'yearly',
      priority: 1,
    },
    {
      url: 'https://acme.com/about',
      lastModified: new Date(),
      changeFrequency: 'monthly',
      priority: 0.8,
    },
    {
      url: 'https://acme.com/blog',
      lastModified: new Date(),
      changeFrequency: 'weekly',
    },
  ]
}
```

ROBOT.TXT

WHAT IS A ROBOTS.TXT FILE?

A robots.txt file tells search engine crawlers which pages or files the crawler can or can't request from your site. The robots.txt file is a web standard file that most good bots consume before requesting anything from a specific domain.



The screenshot shows a web browser window with the address bar displaying `https://kinsta.com/robots.txt`. The page content is as follows:

```
User-agent: *  
Disallow: /wp-admin/  
Allow: /wp-admin/admin-ajax.php  
Sitemap: https://kinsta.com/sitemap_index.xml  
Sitemap: https://kinsta.com/de/sitemap_index.xml  
Sitemap: https://kinsta.com/dk/sitemap_index.xml  
Sitemap: https://kinsta.com/es/sitemap_index.xml  
Sitemap: https://kinsta.com/fr/sitemap_index.xml  
Sitemap: https://kinsta.com/it/sitemap_index.xml
```


PAGES ROUTER

Static robots.txt

```
//public/robots.txt

# Block all crawlers for /accounts

User-agent: \*
Disallow: /accounts

# Allow all crawlers

User-agent: \*
Allow: /
Sitemap: https://example.com/sitemap.xml
```

Learn more about robots.txt in the official Next.js documentation.

Generate a Robots file

```
import type { MetadataRoute } from 'next'

export default function robots(): MetadataRoute.Robots {
  return {
    rules: {
      userAgent: '*',
      allow: '/',
      disallow: '/private/',
    },
    sitemap: 'https://acme.com/sitemap.xml',
  }
}
```

META DATA

PAGE ROUTER

```
import { Html, Head, Main, NextScript } from 'next/document';

export default function Document() {
  return (
    <Html lang='en'>
      <Head>
        <meta charSet='UTF-8' />
        <meta
          name='robots'
          content='index, follow, max-image-preview:large, max-snippet:-1, max-video-preview:-1'
        />
        <meta property='og:locale' content='en_US' />
        <meta name='author' content='Alamin Shaikh' />
        <meta property='og:image:width' content='920' />
        <meta property='og:image:height' content='470' />
        <meta name='twitter:card' content='summary_large_image' />
      </Head>
```

META DATA

APP ROUTER

```
import type { Metadata } from 'next'

// either Static metadata
export const metadata: Metadata = {
  title: 'Next.js',
  description: 'The React Framework for the Web',
  url: 'https://nextjs.org',
  siteName: 'Next.js',
  images: [
    {
      url: 'https://nextjs.org/og.png', // Must be an
      width: 800,
      height: 600,
    },
    {
      url: 'https://nextjs.org/og-alt.png', // Must be
      width: 1800,
```

```
// or Dynamic metadata
import type { Metadata } from 'next'

type Props = {
  params: { id: string }
}

export async function generateMetadata({
  params,
  searchParams,
}: Props): Promise<Metadata> {
  // read route params
  const id = params.id

  // fetch data
  const product = await fetch(`https:// .../${id}`).the
```



THANK YOU!

THANK YOU!