Creating JSON-LD Structured Data in Laravel for SEO

Creating JSON-LD Structured Data in Laravel for SEO

JSON-LD (JavaScript Object Notation for Linked Data) is the recommended way to add structured data to web pages. By embedding JSON-LD in your Laravel app, you can help search engines understand your content and enable rich results like star ratings, breadcrumbs, and knowledge panels. This guide shows how to generate JSON-LD dynamically in Laravel using controllers, Blade templates, and admin fields.

Pass JSON-LD Data from Controller

Each resource (posts, pages, products) can generate its own JSON-LD schema. In your controller, prepare an array and pass it to the view.

```
// app/Http/Controllers/PostController.php
namespace App\Http\Controllers;

use App\Models\Post;

class PostController extends Controller
{
    public function show(Post $post)
    {
        $jsonLd = [
            '@context' => 'https://schema.org',
            '@type' => 'BlogPosting',
            'headline' => $post->title,
```

```
            'author' => [
                '@type' => 'Person',
                'name' => $post->author->name,
            ],
            'datePublished' => $post->created_at->toIso8601String(),
            'dateModified' => $post->updated_at->toIso8601String(),
            'image' => asset('storage/'.$post->cover_image),
            'mainEntityOfPage' => [
                '@type' => 'WebPage',
                '@id' => url('/posts/'.$post->slug),
            ],
        ];

        return view('posts.show', compact('post', 'jsonLd'));
    }
}Code language: PHP (php)
```

The array follows the `BlogPosting` schema. You can customize types like `Product`, `Event`, or `FAQPage` depending on your content.

Render JSON-LD in Blade

In your Blade template, encode the JSON-LD array and output it inside a `<script type="application/ld+json">` tag in the head.

```
<!-- resources/views/posts/show.blade.php -->
@extends('layouts.app')

@section('meta')
  <script type="application/ld+json">
    {!! json_encode($jsonLd, JSON_UNESCAPED_SLASHES|JSON_PRETTY_PRINT)
!!}
  </script>
```

```
@endsection

@section('content')
  <h1>{{ $post->title }}</h1>
  <p>{{ $post->content }}</p>
@endsectionCode language: PHP (php)
```

This ensures that each post page includes structured data that Google and other search engines can parse directly.

UI Example: Admin Schema Fields

Give editors the option to override or extend schema fields from the admin panel. Add inputs for schema type and additional properties:

```
<form method="POST" action="{{ route('posts.store') }}">
  @csrf

  <label>Title</label>
  <input type="text" name="title">

  <label>Meta Description</label>
  <textarea name="meta_description"></textarea>

  <label>Schema Type</label>
  <select name="schema_type">
    <option value="BlogPosting">BlogPosting</option>
    <option value="Product">Product</option>
    <option value="Event">Event</option>
  </select>

  <label>Schema Extra JSON</label>
  <textarea name="schema_json" placeholder='{"ratingValue":
```

```
"5"}'></textarea>

  <button type="submit">Save</button>
</form>Code language: PHP (php)
```

These values can be merged into the JSON-LD array when rendering. For example, if `schema_type` is `Product`, switch the `@type` accordingly.

Common JSON-LD Examples

Here are a few useful schema patterns to include in your Laravel app:

```
{
  "@context": "https://schema.org",
  "@type": "Organization",
  "name": "My Laravel Blog",
  "url": "https://example.com",
  "logo": "https://example.com/logo.png",
  "sameAs": [
    "https://twitter.com/myblog",
    "https://www.linkedin.com/company/myblog"
  ]
}Code language: JSON / JSON with Comments (json)
```

Organization schema builds brand presence in search results.

```
{
  "@context": "https://schema.org",
  "@type": "BreadcrumbList",
  "itemListElement": [
    {
      "@type": "ListItem",
      "position": 1,
```

```
      "name": "Home",
      "item": "https://example.com"
    },
    {
      "@type": "ListItem",
      "position": 2,
      "name": "Blog",
      "item": "https://example.com/blog"
    },
    {
      "@type": "ListItem",
      "position": 3,
      "name": "Laravel JSON-LD",
      "item": "https://example.com/blog/laravel-json-ld"
    }
  ]
}Code language: JSON / JSON with Comments (json)
```

BreadcrumbList schema enhances search results with breadcrumbs instead of long URLs.

Validate Your JSON-LD

After adding JSON-LD, always test with Google's Rich Results Test or the Schema Markup Validator. This ensures your structured data is valid and eligible for enhancements in search results.

Wrapping Up

Adding JSON-LD in Laravel improves SEO and unlocks rich search features. By passing schema arrays from controllers, rendering them in Blade, and giving editors admin UI controls, you make your structured data flexible and scalable. Combine common schemas like BlogPosting, Organization, and BreadcrumbList for maximum impact.

What’s Next

Keep enhancing your SEO stack with these guides:

- Laravel SEO Guide: Optimizing Meta, Slugs, and Sitemaps
- Adding Meta Tags and Open Graph Data Dynamically in Laravel
- How to Build an XML Sitemap Generator in Laravel