Building a Simple Blog with Laravel 12 and Blade

Building a simple blog is a great way to learn Laravel's fundamentals while producing something useful. In this article, we'll walk through setting up a blog with posts, categories, and tags. We'll also cover creating new posts, assigning them to categories and tags, and listing content based on filters. This will give you a clear foundation for building more advanced content-driven applications.

Setting Up Migrations and Models

```
// database/migrations/2025_09_02_000000_create_posts_table.php
Schema::create('posts', function (Blueprint $table) {
    $table->id();
    $table->string('title');
    $table->text('content');
$table->foreignId('category_id')->constrained()->onDelete('cascade');
    $table->timestamps();
});

// database/migrations/2025_09_02_000001_create_categories_table.php
Schema::create('categories', function (Blueprint $table) {
    $table->id();
    $table->string('name');
    $table->string('slug')->unique();
    $table->timestamps();
});

// database/migrations/2025_09_02_000002_create_tags_table.php
Schema::create('tags', function (Blueprint $table) {
    $table->id();
    $table->string('name');
```

```
    $table->string('slug')->unique();
    $table->timestamps();
});

// pivot for posts and tags
Schema::create('post_tag', function (Blueprint $table) {
    $table->foreignId('post_id')->constrained()->onDelete('cascade');
    $table->foreignId('tag_id')->constrained()->onDelete('cascade');
    $table->primary(['post_id', 'tag_id']);
});Code language: PHP (php)
```

We'll use three main entities: `Post`, `Category`, and `Tag`. Posts belong to a category, and can have many tags via a pivot table. This is a classic blog structure.

Defining Relationships in Models

```
// app/Models/Post.php
class Post extends Model
{
    protected $fillable = ['title','content','category_id'];

    public function category()
    {
        return $this->belongsTo(Category::class);
    }

    public function tags()
    {
        return $this->belongsToMany(Tag::class);
    }
}

// app/Models/Category.php
```

```
class Category extends Model
{
    protected $fillable = ['name','slug'];

    public function posts()
    {
        return $this->hasMany(Post::class);
    }
}

// app/Models/Tag.php
class Tag extends Model
{
    protected $fillable = ['name','slug'];

    public function posts()
    {
        return $this->belongsToMany(Post::class);
    }
}Code language: PHP (php)
```

These relationships allow us to fetch a post's category, retrieve posts under a specific category, and query posts by tags.

Creating a Post with Category and Tags

```
// routes/web.php
use App\Http\Controllers\PostController;

Route::resource('posts', PostController::class);

// app/Http/Controllers/PostController.php
class PostController extends Controller
```

```
{
    public function create()
    {
        $categories = Category::all();
        $tags = Tag::all();
        return view('posts.create', compact('categories','tags'));
    }

    public function store(Request $request)
    {
        $validated = $request->validate([
            'title' => 'required|min:3',
            'content' => 'required',
            'category_id' => 'required|exists:categories,id',
            'tags' => 'array'
        ]);

        $post = Post::create($validated);
        $post->tags()->attach($request->input('tags', []));

        return redirect()->route('posts.index')->with('success','Post
created!');
    }
}Code language: PHP (php)
```

Here we load categories and tags into the form, validate user input, create a new post, and attach selected tags. Laravel handles the pivot table automatically.

Blade Template for Post Creation

```
<!-- resources/views/posts/create.blade.php -->
<form action="{{ route('posts.store') }}" method="POST">
    @csrf
```

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

    <label>Content</label>
    <textarea name="content" required></textarea>

    <label>Category</label>
    <select name="category_id">
        @foreach($categories as $category)
            <option value="{{ $category->id }}">{{ $category->name
}}</option>
        @endforeach
    </select>

    <label>Tags</label>
    @foreach($tags as $tag)
        <input type="checkbox" name="tags[]" value="{{ $tag->id }}">
{{ $tag->name }}
    @endforeach

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

This form allows the user to set the title, content, category, and tags when creating a new post. Validation errors and success messages can be added with Blade conditionals.

Listing Posts by Category and Tags

```
// routes/web.php
Route::get('/category/{slug}', [PostController::class, 'byCategory']);
Route::get('/tag/{slug}', [PostController::class, 'byTag']);

// app/Http/Controllers/PostController.php
```

```
public function byCategory($slug)
{
    $category = Category::where('slug',$slug)->firstOrFail();
    $posts = $category->posts()->latest()->get();
    return view('posts.index', compact('posts','category'));
}

public function byTag($slug)
{
    $tag = Tag::where('slug',$slug)->firstOrFail();
    $posts = $tag->posts()->latest()->get();
    return view('posts.index', compact('posts','tag'));
}Code language: PHP (php)
```

These methods let you display posts filtered by category or tag. The `slug` column provides SEO-friendly URLs like `/category/laravel` or `/tag/php`.

```
<!-- resources/views/posts/index.blade.php -->
<h2>Posts</h2>
@foreach($posts as $post)
    <article>
        <h3>{{ $post->title }}</h3>
        <p>Category: {{ $post->category->name }}</p>
        <p>Tags:
            @foreach($post->tags as $tag)
                {{ $tag->name }}
            @endforeach
        </p>
        <p>{{ Str::limit($post->content, 100) }}</p>
    </article>
@endforeachCode language: PHP (php)
```

This Blade view lists posts with their categories and tags. It can be extended with pagination, search, and links to detailed post pages.

Wrapping Up

We've built a simple blog system using Laravel 12 with posts, categories, and tags. Posts are linked to categories and can have multiple tags for flexible content organization. With Blade templates, you can easily create forms for post creation and listing pages filtered by category and tag. This foundation can be expanded into a full CMS with authentication, rich text editors, and SEO optimizations.

What's Next

Now that you can create and filter blog posts, check out these related guides:

- How to Generate SEO-Friendly URLs and Slugs in Laravel
- Adding Meta Tags and Open Graph Data Dynamically in Laravel
- Laravel SEO Guide: Optimizing Meta, Slugs, and Sitemaps