Laravel Project 2: Blog System — Series Ka Imtihan

The capstone: a real blog with posts, categories and comments — every series weapon used once more, in production shape. Series complete.

Introduction

Every series needs a final exam that is not an exam: a real project where the tools stop being lessons and start being reflexes. **Project 2 is a Blog System** — authors write posts, posts live in categories, readers comment. Look at that sentence with LV.1–LV.19 eyes: posts are CRUD (LV.6), authors are authentication (LV.7), categories and comments are relationships (LV.9), the filter is a query chain (LV.10), the pages are components (LV.11), the cover images are uploads (LV.12), the comment guard is middleware (LV.18), and the whole thing earns tests (LV.19) before it deploys (LV.8).

Nothing in this lesson is new. That is precisely the point — and the proof. Build the blog end to end and you have not "finished a course"; you have demonstrated, in one living app, that the full Laravel circle is yours: model, migrate, route, guard, validate, render, relate, test, ship.

(Roman Urdu: Ye imtihan nahi, jashan hai — blog ki har line mein purani bees seekhon ki koi na koi jeet chhupi hai.)

Key points

- **Plan on paper first — three tables:** `posts` (title, body, cover, author, category), `categories` (name), `comments` (body, post, author). The relations draw themselves: a post **belongsTo** an author (User) and a category, and **hasMany** comments; a comment belongsTo its post and its author. Ten minutes of drawing saves ten hours of wandering.
- **Posts are Students Manager in a new coat:** the same resource controller, the same validated forms — plus one grown-up rule: an author edits **only his own posts**. Ownership is checked at the door of `edit` / `update` / `destroy`: not yours → 403 (LV.18's honest wall, now guarding content).
- **The public face is separate from the desk:** visitors read at `/blog` and `/blog/{post}` with no login at all; the writing desk (`/admin/posts`) sits behind `auth`. One app, two doors — the pattern of every CMS on Earth.
- **Category filtering is one where-chain:** `/blog?category=tech` filters through the relation; the category list shows post COUNTS via `withCount('posts')` — LV.9 and LV.10 shaking hands on a public page.
  (Roman Urdu: Category ka filter koi jaadoo nahi — wahi purana where hai, bas rishtay ke raastay chalta hai.)
- **Comments are a guarded mini-CRUD:** only logged-in users comment (guests see "Login to comment"), each comment stamps its author automatically, and the post's author may delete comments on his own post. Create and delete; comments are not edited — like real conversations.

- **Cover images reuse LV.12 whole:** same validation, same random names, same replace-and-cleanup discipline. A skill is only real when it survives a second project without being re-taught.
- **Finish like a professional:** seed the blog with a factory-written welcome post, write the three guard tests (guest cannot post, author edits own, author cannot edit another's), run them green, then deploy by LV.8's five steps. The series ends the way real work ends: tested, shipped, documented.

Code examples

Example 1 — The plan becomes tables and relations

```
// the three migrations, in spirit
Schema::create('categories', function (Blueprint $table) {
    $table->id();
    $table->string('name');
    $table->timestamps();
});

Schema::create('posts', function (Blueprint $table) {
    $table->id();
    $table->foreignId('user_id');       // the author
    $table->foreignId('category_id');
    $table->string('title');
    $table->text('body');
    $table->string('cover')->nullable();
    $table->timestamps();
});

Schema::create('comments', function (Blueprint $table) {
    $table->id();
    $table->foreignId('post_id');
    $table->foreignId('user_id');       // the commenter
    $table->text('body');
    $table->timestamps();
});
```

```
// app/Models/Post.php — the relation web, four sentences
public function author()   { return $this->belongsTo(User::class, 'user_id'); }
public function category() { return $this->belongsTo(Category::class); }
public function comments() { return $this->hasMany(Comment::class); }
```

What it does: Exactly LV.4 and LV.9, composed. `Post::create` in a seeder plants three categories (Tech, School Life, Sports) and a welcome post; the factory writes five practice posts across them.

What you will see: `migrate --seed` ends with a small living blog in the database — and `Post::with(['author', 'category', 'comments'])->get()` in Tinker returns the whole web in 4 queries, eager as LV.9 taught.

Example 2 — Writing desk: store with the author's stamp, edit with the ownership wall

```
public function store(Request $request)
{
    $validated = $request->validate([
        'title'       => 'required|string|max:120',
        'category_id' => 'required|exists:categories,id',
        'body'        => 'required|string',
        'cover'       => 'nullable|image|mimes:jpg,jpeg,png|max:2048',
    ]);

    if ($request->hasFile('cover')) {
        $validated['cover'] = $request->file('cover')->store('covers', 'public');
    }

    $post = $request->user()->posts()->create($validated);

    return redirect()->route('admin.posts.index')
        ->with('success', 'Post published!');
}

public function update(Request $request, Post $post)
{
    if ($post->user_id !== $request->user()->id) {
        abort(403);
    }

    // ... same validation, cover replace with LV.12 cleanup, save, redirect
}
```

What it does: Three reflexes fire together. The author is never a form field — the post is created THROUGH the logged-in user (`$request->user()->posts()`), so authorship cannot be forged. `exists:categories,id` is validation growing up: the chosen category must be real. And update opens with the ownership wall — one comparison, one honest 403, LV.18's spirit in a single line.

What you will see: Fatima publishes "My First Match" under Sports and it appears in her desk list stamped with her name. She opens Imran's post edit address directly — **403**. Her own post edits and republishes with the green flash. The desk is hers; the wall is real.

Example 3 — The public blog: filter, read, comment

```
// the public index — relation filter + counts, eager loaded
public function index(Request $request)
{
    $posts = Post::with(['author', 'category'])
        ->when($request->category, function ($query, $slug) {
            $query->whereHas('category', fn ($q) => $q->where('slug', $slug));
        })
        ->latest()
        ->paginate(6);

    $categories = Category::withCount('posts')->get();

    return view('blog.index', compact('posts', 'categories'));
}
```

```
// comments — logged-in hands only, author stamped by the relation
public function storeComment(Request $request, Post $post)
{
    $request->validate(['body' => 'required|string|max:500']);

    $post->comments()->create([
        'body'    => $request->body,
        'user_id' => $request->user()->id,
    ]);

    return back()->with('success', 'Comment posted!');
}
```

What it does: The public page composes the deep lessons: `with()` keeps the query count flat (LV.9), `whereHas` filters by a relation's field (LV.10), `paginate(6)` serves six posts per page, and `withCount` powers the sidebar's "Tech (4)" labels without extra loops. The comment route sits behind `auth`; the comment's author is stamped from the session user, never from the form.

What you will see: `/blog` shows six cards with author and category lines; clicking **Tech (4)** filters to four posts; a post page lists its comments with names; a guest scrolling down finds *Login to comment* instead of a form, while the teacher posts a comment and sees it land with her name and the green flash. Every weapon, one page.

Quick recap

- The blog is the series in one app: three tables, a relation web, guarded CRUD, ownership walls, eager public pages.

- Authorship is stamped through relations, never through form fields; ownership is checked before any edit lands.
- Public reading and the private desk are two doors of one app — guests read, members write, authors rule their own posts.
- Tested green, then shipped by the five steps: the Laravel circle — model, migrate, route, guard, validate, render, relate, test, ship — is now yours.

Common mistakes

1. **Trusting a hidden author field** — a form carrying `user_id` (even hidden) lets anyone publish as anyone by editing the request. Fix: the author comes from `$request->user()` through the relation; identities are never inputs.
2. **Skipping the ownership check "because the link is hidden"** — the edit URL is guessable (`/posts/7/edit`), and hidden is not guarded (LV.7/LV.18's recurring law). Fix: the 403 wall inside update/destroy, tested by LV.19's robot.
3. **The blog index that melts at 100 posts** — `Post::all()` plus lazy author/category per card is LV.9's N+1 wearing a public face. Fix: `with()` + `paginate()` from the FIRST version — public pages are where traffic judges you.
4. **Calling it done without the guard tests** — a capstone shipped on hand-clicks alone forgets LV.19 the moment it matters most. Fix: the three promises (guest, own, other's) run green before the deploy ritual begins.

Practice

1. Add the reading-time kindness: a `reading_time` line on the post page computed from the body (`str_word_count` ÷ 200, rounded up, "3 min read"). Then cover it with a unit test using a body of exactly 400 words expecting 2.
2. Grow one branch of your choice, properly: post search by title (a `where('title', 'like', ...)` chain), OR scheduled publishing (a `published_at` column + a scope that hides future posts from the public index). Whichever you choose: migration, relation/scope, page, and one feature test — the full circle, once more, alone.

Practice quiz

1. In the blog project, what are the two doors of the one app?

- The home page and the login page
- Visitors read at /blog and /blog/{post} with no login at all; the writing desk at /admin/posts sits behind auth
- The API and the website
- The desktop site and the mobile site

Answer: Visitors read at /blog and /blog/{post} with no login at all; the writing desk at /admin/posts sits behind auth

2. Who may edit a blog post, and how is that enforced?

- Anyone who finds the edit link
- Only its own author: ownership is checked at the door of edit, update and destroy, and not yours means 403
- Any logged-in user
- Only admins, authors may only read

Answer: Only its own author: ownership is checked at the door of edit, update and destroy, and not yours means 403

3. Where must a post's author come from, and why not from a form field?

- From a hidden user_id field, it is convenient
- From $request->user() through the relation; a form carrying user_id, even hidden, lets anyone publish as anyone by editing the request
- From the URL
- Authors are not recorded

Answer: From $request->user() through the relation; a form carrying user_id, even hidden, lets anyone publish as anyone by editing the request

---

Watch the Code Scope channel: https://www.youtube.com/@CodeScopePK — this lesson's own video joins the channel when it is recorded.

Read online: https://scopetreasure.com/notes/laravel-blog-project · More free tools and notes at https://scopetreasure.com