Laravel CRUD Project — Students Manager App

The whole toolkit in one app: Route::resource and its 7 routes, create-read-update-delete live, method spoofing and green success messages.

Introduction

Everything so far — routes, Blade, models, forms — was practice for this lesson: a complete little system. **CRUD** is the four jobs every app on Earth does with data: **Create, Read, Update, Delete**. Students Manager does all four for our five students, and it does them the way real Laravel apps do: one controller, seven standard routes, forms for create and edit, a confirm-guarded delete, and a green success message after every change.

Laravel even has a single line that wires the whole pattern: `Route::resource`. Learn this lesson properly and you have not just "done a tutorial project" — you own the skeleton of every admin panel, shop, blog and booking system you will ever build. They are all this, wearing different clothes.

(Roman Urdu: CRUD har app ki rooh hai — banana, parhna, badalna, mitana. Ye chaar kaam aa gaye to har system banana aa gaya.)

Key points

- **`Route::resource('students', StudentController::class)`** creates the 7 standard routes in one line: index (list), create (form), store (save new), show (one), edit (form), update (save changes), destroy (delete). Memorise the seven names — the whole Laravel world uses them.
- **Route names follow the pattern:** `students.index`, `students.create`, `students.store`, `students.show`, `students.edit`, `students.update`, `students.destroy` — so `route('students.edit', 5)` is the edit page of student 5, everywhere, forever.
- **HTML forms only speak GET and POST** — so Laravel fakes the other verbs: `@method('PUT')` inside the edit form and `@method('DELETE')` inside the delete form whisper the real verb to Laravel. This is called method spoofing, and it is completely standard.
- **Route model binding keeps methods short:** type-hint `Student $student` in `update(Request $request, Student $student)` and Laravel fetches the student from the address id itself — 404 included, free.
- **Flash messages ride the session:** `->with('success', 'Student added!')` stores a one-request message; Blade shows it once with `session('success')` and it vanishes. (Sessions return in depth at LV.17.)

  (Roman Urdu: Flash message mehman hai jo aik hi baar salaam karta hai — agli refresh par ghayab.)
- **Delete is a form, never a link:** a GET link can be triggered by crawlers and accidents. Destroy is always a small POST form with `@method('DELETE')` and a confirm question.

- **The controller stays skinny:** each method does one job — validate, touch the model, redirect with a message. If a method grows past a screen, it is doing someone else's job.

Code examples

Example 1 — One line, seven routes

```
// routes/web.php
Route::resource('students', StudentController::class);
```

```
php artisan route:list --name=students
```

What it does: `Route::resource` reads the controller and registers the standard seven. The `route:list` X-ray proves nothing is missing and shows exactly which method answers each address.

What you will see:

```
  GET|HEAD        students ............ students.index › StudentController@index
  GET|HEAD        students/create ... students.create › StudentController@create
  POST            students ............ students.store › StudentController@store
  GET|HEAD        students/{student} .. students.show › StudentController@show
  GET|HEAD        students/{student}/edit .. students.edit › StudentController@edit
  PUT|PATCH       students/{student} .. students.update › StudentController@update
  DELETE          students/{student} .. students.destroy › StudentController@destroy
```

Example 2 — Edit + update: prefilled form, spoofed verb, bound model

```
{{-- students/edit.blade.php — the LV.5 form, prefilled --}}
<form method="POST" action="{{ route('students.update', $student) }}">
    @csrf
    @method('PUT')

    <input type="text" name="name" value="{{ old('name', $student->name) }}">
    <input type="text" name="marks" value="{{ old('marks', $student->marks) }}">
    <button type="submit">Update Student</button>
</form>
```

```
// StudentController
public function update(Request $request, Student $student)
{
    $validated = $request->validate([
        'name'  => 'required|string|max:50',
        'marks' => 'required|integer|min:0|max:100',
    ]);

    $student->update($validated);

    return redirect()->route('students.index')
        ->with('success', 'Student updated!');
}
```

What it does: Three ideas meet here. `@method('PUT')` upgrades the POST form to a PUT request. `old('marks', $student->marks)` is genius in a small way: show the typed value if the user just bounced, otherwise the saved value. And route model binding hands `update()` the ready student — no `find()` line needed. `$student->update($validated)` writes the changes (the model needs the fields listed in `$fillable` for this mass-assignment style — Laravel's safety lock against unwanted fields, worth meeting now).

What you will see: open student 3 (Bilal, 76), change marks to `90`, submit — the list greets you with a green `Student updated!` banner and Bilal's row now reads 90.

Example 3 — Delete with a guard, and the list that shows the flash

```
{{-- inside the index table, one small form per row --}}
<form method="POST" action="{{ route('students.destroy', $student) }}"
      onsubmit="return confirm('Delete this student?')">
    @csrf
    @method('DELETE')
    <button type="submit">Delete</button>
</form>
```

```
public function destroy(Student $student)
{
    $student->delete();

    return redirect()->route('students.index')
        ->with('success', 'Student deleted!');
}
```

```
{{-- students/index.blade.php — the flash banner, above the table --}}
@if (session('success'))
    <p class="success">{{ session('success') }}</p>
@endif
```

What it does: Delete asks first (`confirm`), then submits a real DELETE request. The controller deletes through the bound model and redirects with a flash. The index page checks the session once and shows the green banner for exactly one page load.

What you will see: click Delete on a practice row, confirm — the row is gone, `Student deleted!` glows green at the top, and on refresh the banner is gone too. Flash kept its promise: one visit only.

Quick recap

- CRUD = Create, Read, Update, Delete — and `Route::resource` wires all 7 standard routes in one line.
- Edit/delete forms spoof PUT/DELETE with `@method`; route model binding delivers the model ready-made.
- Every change ends the same way: redirect to the list with a one-time flash success message.
- Delete is always a guarded form, never a naked link.

Common mistakes

1. **A delete LINK (`<a href="/students/5/delete">`)** — GET requests must never change data; one crawler visit can empty your table. Fix: destroy is a POST form with `@method('DELETE')` and a confirm, always.
2. **`$student->update()` silently doing nothing** — mass assignment is locked until fields are listed. Fix: add `protected $fillable = ['name', 'city', 'marks'];` to the model (LV.4's class grows this line now).
3. **Returning a view after store/update** — the user sits on the POST result and a refresh duplicates the record. Fix: writes always end in `redirect()` (the Post/Redirect/Get pattern) — LV.5's rule, now law.
4. **Validating create but not update** — the edit form is the same door; unguarded, it saves the same bad data. Fix: both `store()` and `update()` validate first, with the same rules.

Practice

1. Finish the full Students Manager: all seven methods, the index table with Edit and Delete per row, and flash banners. Add the five series students and show a class average (83) under the table, computed from the collection.
2. Add a "City" edit to the update flow end-to-end (form field, rule, fillable) and move one student from Sahiwal to Multan. Then break the update deliberately — marks `500` — and confirm the bounce keeps your typed values.

Practice quiz

1. How many routes does Route::resource('students', StudentController::class) create, and what are they?

- 5: index, store, show, update, destroy
- 7 in one line: index (list), create (form), store (save new), show (one), edit (form), update (save changes), destroy (delete)
- 2: index and show
- 7, but all of them are GET routes

Answer: 7 in one line: index (list), create (form), store (save new), show (one), edit (form), update (save changes), destroy (delete)

2. Why do edit and delete forms carry @method('PUT') and @method('DELETE')?

- Because Laravel requires decoration on every form
- Because HTML forms only speak GET and POST, so Laravel fakes the other verbs with method spoofing, which is completely standard
- Because browsers cannot send POST
- To make the form faster

Answer: Because HTML forms only speak GET and POST, so Laravel fakes the other verbs with method spoofing, which is completely standard

3. Why must delete be a form, never a link?

- Links are slower
- A GET link can be triggered by crawlers and accidents; destroy is always a small POST form with @method('DELETE') and a confirm question
- Links cannot carry an id
- Forms look better

Answer: A GET link can be triggered by crawlers and accidents; destroy is always a small POST form with @method('DELETE') and a confirm question

---

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-crud-project · More free tools and notes at https://scopetreasure.com