ScopeTreasure Courses

LARAVEL COURSES · LESSON LV.17 · CODE SCOPE SERIES

Sessions & Cookies Deep — Website Ki Yaadein

The website's two memory boxes: cookies live in the browser, sessions live on the server behind one key cookie — put/get/flash, drivers and the sessions table.

Introduction

HTTP has no memory: every request arrives like a stranger. Yet your app *knows* you are logged in (LV.7), remembered your flash message for exactly one page (LV.6), and greets you by name. Two memory boxes do this work, and confusing them causes real bugs — this lesson finally opens both.

A **cookie** is a small note the server asks the BROWSER to keep: "this visitor's language is Urdu". A **session** is a store on the SERVER — a whole drawer per visitor — and the browser holds only its KEY: one cookie carrying the session id. Login state, carts and flash messages live in the drawer, never in the browser. Cookies: the key on your ring. Sessions: the house it opens.

(Roman Urdu: Cookie chaabi hai jo mehman ke paas rehti hai; session woh ghar hai jo server par hai — data ghar mein, chaabi jeb mein.)

Key points

- **The cookie flow is a header conversation:** the server answers with `Set-Cookie: visited=yes`, the browser stores it, and every later request quietly attaches `Cookie: visited=yes`. You read it back with `$request->cookie('visited')` — no database, no storage of yours involved.
- **Cookies live in the browser — act accordingly:** the visitor (and any script that reaches them) can READ them. Cookies hold preferences and keys, never secrets: no passwords, no marks, no "is_admin=true" flags a user could simply edit.
- **Sessions are server-side drawers:** `session()->put('key', $value)`, `->get('key')`, `->has('key')`, `->forget('key')`. The data sits on YOUR machine; the browser's only cookie is `laravel_session` — a long random id that means nothing by itself. Steal it and you hold a key, not the contents.
- **Flash is a session with a lifespan:** LV.6's `->with('success', ...)` stores a value marked "delete after the NEXT request" — one greeting, then gone. Anything that must survive two page loads does not belong in flash.

  (Roman Urdu: Flash woh mehman hai jo salaam karke nikal jata hai; jo do safay rukay, woh session mein rehta hai, flash mein nahi.)
- **Remember me is a long-life key:** LV.7's checkbox plants a special long-lived cookie so the browser can re-open the session drawer weeks later without a password. Convenience and risk in one cookie — on shared computers it outlives your visit.
- **Drivers decide where drawers live:** `SESSION_DRIVER` in `.env` — `file` (drawer = files in `storage/framework/sessions`, the default), `cookie` (the whole drawer travels, encrypted, in the cookie),

`database` (rows in a **sessions** table — Laravel 11's default migration already creates it), `redis` (the fast production drawer). Changing drivers is one `.env` line.

- **The sessions table tells the truth:** with the database driver, one row per visitor: id, `user_id` (filled after login!), IP address, and a `payload` — the drawer's contents, serialised and base64'd. Readable structure, unreadable secrets — as it should be.

Code examples

Example 1 — A plain cookie: set it, read it back, retire it

```
// a route that plants the note
Route::get('/set-visit', function () {
    return response('Welcome!')
        ->cookie('visited', 'yes', 60);   // name, value, minutes
});

// any later route reads the ring
Route::get('/check-visit', function (Request $request) {
    return $request->cookie('visited') === 'yes'
        ? 'Welcome back, we remember your visit.'
        : 'First time here?';
});

// retiring a cookie = sending it already expired
use Illuminate\Support\Facades\Cookie;

Route::get('/forget-visit', function () {
    return response('Forgotten.')->withCookie(Cookie::forget('visited'));
});
```

What it does: The first response carries the `Set-Cookie` header; the browser obeys silently and attaches the note to every later request for an hour. Reading is one method call. Forgetting is not deleting on the server — there is nothing ON the server; you ask the browser to drop its note by re-issuing it expired.

What you will see: visit `/set-visit`, then `/check-visit` — *Welcome back* — and the browser's developer tools (Application → Cookies) show the small row: `visited | yes`. After `/forget-visit`, the row is gone and the check answers *First time here?*

Example 2 — The session drawer: "last viewed student"

```
// StudentController@show — remember the visit in the drawer
public function show(Student $student)
{
    session()->put('last_student', $student->name);

    return view('students.show', compact('student'));
}

// dashboard Blade — the drawer speaks
@if (session()->has('last_student'))
    <p>You last viewed: {{ session()->get('last_student') }}</p>
@endif

// and leaving no trace when asked
session()->forget('last_student');
```

What it does: Viewing Hira stores her NAME in the server-side session — the browser receives no new information at all; its `laravel_session` key cookie does not even change. The dashboard asks the drawer, not the browser. `forget()` empties one slot; the drawer and the login survive.

What you will see: open Hira's page, return to the dashboard — *You last viewed: Hira Shah*. Open the browser's cookies to check what travelled: only the session id, exactly as promised. Log out and back in — the slot is empty, because a new drawer begins with the new login.

Example 3 — Changing drivers and peeking into the sessions table

```
# .env — drawers move from files into the database
SESSION_DRIVER=database
```

What it does: Modern Laravel's default tables migration (the same one that brought jobs in LV.16) already built the `sessions` table, so the switch is truly one line. Log in as the teacher and visit a few pages, then open the table.

What you will see: one row per visitor — and the moment you log in, `user_id` fills with the teacher's id, quietly proving which drawer belongs to whom. The `payload` column holds the drawer's contents as a long base64 string (flash, login state, last_student) — structure visible to you, contents sealed from casual eyes. Switch back to `SESSION_DRIVER=file` and the same drawers reappear as small files under `storage/framework/sessions/` — same memory, different furniture.

Quick recap

- Cookies: small notes in the browser (preferences + keys only — readable, so never secrets).

- Sessions: drawers on the server; the browser holds only the session-id cookie. `put/get/has/forget`, with flash as the one-request cousin.
- `SESSION_DRIVER` (file / cookie / database / redis) chooses where drawers live; the database driver's table shows user_id and payload per visitor.

Common mistakes

1. **Secrets in cookies** — a cookie holding marks, roles or any value the app then TRUSTS. The user can edit cookies in seconds. Fix: cookies carry preferences and keys; every trusted fact lives server-side in the session/database.
2. **A password in the session "for convenience"** — the session payload is storage you do not fully control (files, shared tables). Fix: sessions hold ids and state; passwords exist only as LV.7 hashes, nowhere else, ever.
3. **Expecting flash to wait** — storing "the thing the user needs on the page after next" with `with()` and watching it vanish after one hop (two redirects eat it). Fix: flash is for the very next screen's greeting; anything longer gets a normal `put()`.
4. **"Session lost" in incognito confusion** — closing the private window deletes the browser's cookies, so the drawer key is gone though the drawer exists. Fix: that IS the design — no key, no entry; test logins in a normal window when persistence is what you are checking.

Practice

1. Build a language preference properly: a `/lang/urdu` route plants a `lang` cookie for 30 days; the layout greets in that language on every page. Then edit the cookie's value by hand in dev tools to `xx` and make the app fall back gracefully — write the one line that guards it.
2. Add a "recently viewed" list to the session: each `show()` pushes the student's name onto a `recent` array (keep at most 3). Print it on the dashboard and verify it slides correctly as you view Ali, Sara, Hira, then Usman.

Practice quiz

1. What lives in a cookie, and what must never live there?

- Everything, cookies are private
- Cookies live in the browser and the visitor can read them, so they hold preferences and keys only, never secrets: no passwords, no marks, no is_admin=true flags a user could simply edit
- Sessions live in cookies
- Passwords live in cookies safely

Answer: Cookies live in the browser and the visitor can read them, so they hold preferences and keys only, never secrets: no passwords, no marks, no is_admin=true flags a user could simply edit

2. Where does session data live, and what does the browser hold?

- In the browser, as a cookie
- On the server, in drawers; the browser's only cookie is laravel_session, a long random id that means nothing by itself
- In the database only, always
- Nowhere, sessions are imaginary

Answer: On the server, in drawers; the browser's only cookie is laravel_session, a long random id that means nothing by itself

3. How long does a flash message live?

- Forever, until deleted
- One greeting, then gone: it is marked delete after the NEXT request, so anything that must survive two page loads does not belong in flash
- Until the user logs out
- One hour

Answer: One greeting, then gone: it is marked delete after the NEXT request, so anything that must survive two page loads does not belong in flash

---

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-sessions-cookies · More free tools and notes at https://scopetreasure.com