

**CARA DEBUGGING  
(LARAGON)**



**Disusun Oleh  
ABDULLAH AZZAM RABBANI  
10240038**

**PROGRAM STUDI REKAYASA PERANGKAT LUNAK  
UNIVERSITAS BINA SARANA INFORMATIKA  
MARGONDA  
2025/2026**

## 1. Implementasi Debugging Dengan Dd() Atau Ddd()

Kriteria ini fokus pada inspeksi data yang masuk (\$request) saat terjadi operasi seperti Create (Store) atau Update.

### Skenario: Debugging Data Input (Store/Create)

Misalnya Anda memiliki *Controller* untuk mengelola data anggota (AnggotaController). Anda ingin memeriksa data apa saja yang dikirimkan pengguna sebelum melakukan validasi atau penyimpanan.

```
class AnggotaController extends Controller
{
    /**
     * Store a newly created resource in storage.
     */
    public function store(Request $request)
    {
        // Contoh dengan dd() untuk memenuhi kriteria
        dd($request->all()); // Mencetak semua data input dan menghentikan proses [cite: 1, 3]

        // --- KODE BERIKUT TIDAK AKAN PERNAH TERCAPAI KARENA dd() MENGHENTIKAN EKSEKUSI ---

        $validatedData = $request->validate([
            'nama' => 'required|max:255',
            'hp' => 'required|min:10|max:13',
        ]);

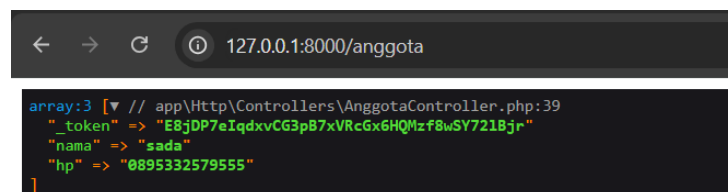
        if (!is_array($validatedData)) {
            throw new \Exception('message: "Validated data is not an array!');
        }

        Anggota::create($validatedData);
        return redirect('to: '/anggota');
    }
}
```

Gambar 1: contoh menggunakan dd(\$request) pada fuction store

### Pemeriksaan di Browser (dengan Laragon)

1. Jalankan Laragon dan pastikan proyek Laravel Anda dapat diakses.
2. Akses formulir *Create* Anda dan kirimkan data.
3. Alih-alih diarahkan ke halaman berikutnya, browser akan menampilkan halaman debugging yang berisi struktur data lengkap dari array input (nama produk, harga, token CSRF, dll.) yang diterima oleh \$request->all().



```
array:3 [▼ // app\Http\Controllers\AnggotaController.php:39
  "_token" => "E8jDP7eIqdxvCG3pB7xVRcGx6HQMzf8wSY721Bjr"
  "nama" => "sada"
  "hp" => "0895332579555"
]
```

Gambar 1.1: Hasil running browser setelah debug

Jika sudah tidak memerlukan debugging ini, anda bisa menghapus bari dd(\$request) atau menkommentarnya.

## 2. Implementasi Invalid-Feedback Pada Form Validasi

Kriteria ini fokus pada cara Laravel dan Bootstrap (umumnya digunakan) menampilkan pesan kesalahan di browser dan bagaimana cara menyelesaikannya.

### Skenario: Menangani Validasi Input yang Gagal

Anda ingin memastikan pengguna melihat pesan kesalahan yang jelas di bawah kolom input jika mereka gagal mengisi formulir dengan benar (misalnya, nama wajib diisi).

#### Controller (Penanganan Validasi)

Pastikan Anda menjalankan validasi. Jika gagal, Laravel secara otomatis akan mengarahkan pengguna kembali dan menyimpan kesalahan di *session* \$errors.

```
public function store(Request $request)
{
    // Validasi data input. Jika gagal, exception dilempar, dan pengguna dikembalikan ke form.
    $validatedData = $request->validate([
        'nama' => 'required|max:255', // Aturan: Nama produk harus diisi
        'hp' => 'required|min:10|max:13',
    ]);
    if (!is_array($validatedData)) {
        throw new \Exception('Validated data is not an array!');
    }
    Anggota::create($validatedData);
    return redirect(to: '/anggota');
}
```

Gambar 1.2: validasi pada kolom nama dan hp

#### View (Penyelesaian error/Penanganan pesan error)

Di file blade (resources/views/v\_anggota/create.blade.php), Anda harus menggunakan directive @error untuk menampilkan pesan error bootstrap .invalid-feedback

```
<h3> {{{ $judul }}} </h3>
<form action="{{ route('anggota.store') }}" method="post">
    @csrf
    <label>Nama</label><br>
    <input type="text" name="nama" id="" value="{{ old('nama') }}" placeholder="Masukkan Nama Lengkap" class="form-control @error('nama') is-invalid @enderror">
    {{{-- Kriteria 2: invalid-feedback (Penanganan pesan error pada browser) --}}}
    @error('nama')
    <span class="invalid-feedback alert-danger" role="alert">
        {{{ $message }}} {{{-- Menampilkan pesan kesalahan spesifik untuk field 'nama' --}}}
    </span>
    @enderror
</form>
```

Gambar 1.3: Contoh validasi error pada field nama

The screenshot shows a web browser address bar with the URL `127.0.0.1:8000/anggota/create`. Below the address bar is a heading **Tambah Anggota**. There are two form fields: 'Nama' with a placeholder 'Masukkan NamaLengkap' and 'HP' with a placeholder 'Masukkan NomorHP'. Both fields have red error messages: 'The nama field is required.' and 'The hp field is required.' respectively. At the bottom of the form are two buttons: 'Simpan' and 'Batal'.

**Gambar 1.4:** tangkapan layer hasil form yang sudah di validasi

### Cara Penyelesaian Masalah Validasi:

1. Masalah (Pesan error Tidak Muncul): Saat validasi gagal (misalnya, kolom nama kosong), pesan error tidak terlihat di browser.
2. Penyebab: Kelas `invalid-feedback` di Bootstrap disembunyikan secara default (`display: none`). Ia hanya akan ditampilkan ketika elemen input yang berada tepat sebelum atau sejajar dengannya memiliki kelas `is-invalid`.
3. Penyelesaian (Debugging): Tambahkan Blade *directive* `@error('field_name')` `is-invalid` `@enderror` pada elemen `<input>` yang bersangkutan. Ini akan secara kondisional menambahkan kelas CSS `is-invalid` ke `<input>` hanya ketika ada kesalahan validasi untuk *field* tersebut, sehingga **invalid-feedback** di bawahnya akan muncul di browser.