

Laravel 13 validation cheat sheet

Where to validate, the typed Rule:: builders that arrived in 13.36, conditional and array rules, database, date, file and password rules, custom rules, and how the errors reach a Vue form with Inertia and Precognition. Every chain was run on Laravel 13.37.



Laravel 13.37 · Inertia 3

Where to validate

In the controller

```
$validated = $request->validate([
    'title' => ['bail', 'required', 'string', 'max:255'], //
    // bail: stop this field at the first failure
    'body' => ['required'],
]);
$request->validated(); // $request->safe()-
>only(['title']); $request->safe()->except(['body']);
$request->validateWithBag('post', [...]);
// named error bag
```

Form Request (php artisan make:request StorePostRequest)

```
public function authorize(): bool { return $this->user()-
>can('create', Post::class); }
public function rules(): array { return ['title' =>
[Rule::required(), Rule::string()->max(255)]]; }
public function messages(): array { return ['title.required'
=> 'Give the post a title.']; }
public function attributes(): array { return ['email' => 'email
address']; } // :attribute in messages
public function after(): array { return [fn (Validator $v)
=> $v->errors()->add('title', 'Nope.')] }; }
// Manual: Validator::make($data, $rules)->after(fn ($v) =>
...)->stopOnFirstFailure()->validate();
```

Rule:: builders (13.36): typed instead of strings

Autocomplete, and comma-safe values

```
use Illuminate\Validation\Rule;

'account_type' => [Rule::required(), Rule::in(['personal',
'business'])],
'company' => [Rule::missingIf('account_type',
'personal')],
'email' => [Rule::requiredWithout('phone'),
Rule::email()],
'phone' => [Rule::requiredWithout('email'),
Rule::regex('/^\+?[0-9 ]{7,15}$')],
'password' => [Rule::required(), Rule::confirmed()-
>customField('password_repeat')],
'server_ip' => [Rule::nullable(), Rule::ip()->version(4)],
// it is ip(), not ipAddress()
'terms' => [Rule::accepted()],
(string) Rule::requiredWith('phone', 'email'); //
"required_with:phone,email": strings still work, mix freely
```

Strings, numbers, booleans, enums

Fluent type rules

```
'title' => [Rule::string()->min(3)->max(255)],
// alpha(), alphaDash(), ascii(), startsWith(), lowercase()...
'slug' => [Rule::regex('/^[a-z0-9-]+$')], Rule::notRegex('/-
-/' ),
'price' => [Rule::numeric(strict: true)->min(0)->decimal(0,
2)], // strict (13.37): "9.5" fails, 9.5 passes
'qty' => [Rule::numeric()->integer()->between(1, 100)],
// digits(), maxDigits(), multipleOf(), greaterThan('min')
'tos' => [Rule::boolean(strict: true)],
// only true/false, not "1"/"true"
'status' => [Rule::enum(Status::class)->only([Status::Draft,
Status::Live])], // except()
'color' => [Rule::hexColor()], 'meta' => [Rule::json()],
'tz' => [Rule::timezone()], 'id' => [Rule::uuid()->version(7)]
```

Conditional rules

Field-based, callback-based, and sometimes

```
'company' => [Rule::requiredIf(fn () => $this-
>input('account_type') === 'business')], // requiredUnless,
prohibitedIf, excludeIf: callback or bool
'vat' => [Rule::presentIf('account_type', 'business'),
Rule::nullable()], // missingIf, acceptedIf,
declinedIf: field + value
'notes' => [Rule::prohibits('tagline')], 'legacy' =>
[Rule::excludeWith('new_field')], // excluded from validated()
'plan' => Rule::when($this->user()->isPro(), ['required',
'in:team,business'], ['nullable']), // Rule::unless(...)
'reason' => ['sometimes', 'required', 'max:500'], // only
validated when present in the input

$validator->sometimes('reason', ['required', 'max:500'], fn
(Fluent $input) => $input->games >= 100);
```

Arrays and nested data

Wildcards, keys, per-item rules

```
'tags' => [Rule::array(), 'max:5'],
'tags.*' => [Rule::string()->max(20), Rule::distinct()-
>ignoreCase()], // error lands on tags.1
'address' => [Rule::array(['street', 'city'])],
// only these keys allowed; requiredArrayKeys()
'options' => [Rule::contains(['a', 'b'])],
// doesntContain()
'companies.*.id' => Rule::forEach(fn (?string $value, string
$attribute) => [
Rule::exists(Company::class, 'id'), new
HasPermission('manage-company', $value),
]),
'username' => [Rule::required(), Rule::anyOf(['string',
'email'], ['string', 'alpha_dash', 'min:6'])],
```

Database, dates, formats

unique, exists, date, email, url

```
'email' => [Rule::unique('users', 'email')->ignore($this-
>user())->withoutTrashed()], // ignoreModel(), whereNull(),
using(fn ($q) => ...)
'team' => [Rule::exists('teams', 'id')->where(fn ($q) => $q-
>where('owner_id', $this->user()->id)]],
'publish_at' => [Rule::date()->format('Y-m-d')->afterToday()],
// before(), between($a, $b), todayOrAfter(), equals('2026-10-
14')
'email' => [Rule::email()->rfcCompliant(strict: true)-
>preventSpoofing()->validateMxRecord()],
'site' => [Rule::url()->protocols(['https'])], 'ip' =>
[Rule::ip()->version(6)], 'code' => [Rule::ulid()],
```

Files and passwords

File::image, File::types, Password

```
use Illuminate\Validation\Rules\{File, Password};

'cover' => [Rule::nullable(), File::image()->min(1024)->max(12
* 1024) // KB; or ->max('12mb')
->dimensions(Rule::dimensions()->maxWidth(2000)->ratio(16 /
9))], // minWidth(), ratioBetween()
'report' => [File::types(['pdf', 'csv'])->max('10mb')],
// Rule::imageFile(allowSvg: true)
'password' => [Rule::required(), Rule::confirmed(),
Password::min(8)->letters()->mixedCase()->numbers()->symbols()-
>uncompromised()],
'current_password' => [Rule::currentPassword()],
// AppServiceProvider::boot(): Password::defaults(fn () =>
$this->app->isProduction() ? Password::min(12)->uncompromised()
: Password::min(8));
'password' => ['required', Password::defaults()],
```

Custom rules

make:rule, closures, implicit

```
php artisan make:rule Uppercase # --implicit to run
on empty values too

class Uppercase implements ValidationRule // DataAwareRule to
see other fields, ValidatorAwareRule for the validator
{
    public function validate(string $attribute, mixed $value,
Closure $fail): void
    {
        if (strtoupper($value) !== $value) $fail('The
:attribute must be uppercase.');
```

```
    }
}
'name' => ['required', new Uppercase, fn (string $attribute,
mixed $value, Closure $fail) => $value === 'foo' && $fail("
{$attribute} is invalid.")],
```

Errors in Vue with Inertia

useForm, <Form> and Precognition

```
<script setup lang="ts">
import { useForm } from '@inertiajs/vue3'
const form = useForm({ title: '', email: '' })
const submit = () => form.post('/posts', { onError: (errors) =>
console.log(errors) }) // 422 → form.errors, no page change
</script>
<template>
    <form @submit.prevent="submit">
        <input v-model="form.title"
@input="form.clearErrors('title')" />
        <p v-if="form.errors.title">{{ form.errors.title }}</p>
        <!-- first message per field -->
        <button :disabled="form.processing">Save</button>
    </form>
    <!-- Live validation with the same rules (Precognition
middleware on the route): -->
    <Form action="/posts" method="post" #default="{ errors,
invalid, validate, validating }">
        <input name="title" @change="validate('title')" /> <p v-
if="invalid('title')">{{ errors.title }}</p>
    </Form>
</template>
<!-- JSON clients get 422 { message, errors: { title: ['...'] }
} -->
```

Checked 2026-10-17 against: <https://laravel.com/docs/13.x/validation> · <https://github.com/laravel/framework/blob/13.x/src/Illuminate/Validation/Rule.php> · <https://github.com/laravel/framework/pull/61833> · <https://inertiajs.com/docs/v3/the-basics/forms>

Online version with copy buttons: [laravel-vuejs.space/cheatsheets/laravel-validation](#) · More guides and the monthly newsletter at [laravel-vuejs.space](#)