ValidationError

August 17, 2026

ValidationError

Kind: Interface

Source: `packages/common/interfaces/external/validation-error.interface.ts`

Part of: Common

Validation error description.

`ValidationError` describes a failed validation result for a single property on an object. It captures the validated target, invalid value, violated constraints, optional validation contexts, and nested errors for child objects or arrays.

Properties

Property	Type
`target`	`Record<string, any>`
`property`	`string`
`value`	`any`
`constraints`	`{ [type: string]: string; }`
`children`	`ValidationError[]`
`contexts`	`{ [type: string]: any; }`

Diagram

```
graph LR
  A[ValidationError] --> B[target: Record]
  A --> C[property: string]
  A --> D[value: any]
  A --> E[constraints: constraint messages]
  A --> F[contexts: constraint metadata]
  A --> G[children: ValidationError[]]
```

```
  G --> H[Nested property errors]
  E --> I[Validation failure messages]
```

Usage

```ts
import type { ValidationError } from '@your-package/common';

const error: ValidationError = {
  target: {
    email: 'invalid-email',
  },
  property: 'email',
  value: 'invalid-email',
  constraints: {
    isEmail: 'email must be a valid email address',
  },
  contexts: {
    isEmail: {
      code: 'INVALID_EMAIL',
    },
  },
  children: [],
};

function formatValidationError(validationError: ValidationError): string[] {
  const messages = Object.values(validationError.constraints ?? {});

  return [
    ...messages,
    ...validationError.children.flatMap(formatValidationError),
  ];
}

console.log(formatValidationError(error));
// ['email must be a valid email address']
```

AI Coding Instructions

- Preserve nested validation failures in `children`; do not discard them when formatting errors for nested DTOs or array elements.
- Treat `constraints` and `contexts` as potentially absent or empty when consuming validation results.
- Use `property` together with parent error paths to build complete field paths such as `address.street`.

- Avoid exposing `target` or raw `value` directly in API responses when they may contain sensitive data.
- Keep constraint keys aligned with the validation library or custom validator names that produced the error.