

Introduction

Validate, convert, interpret and inspect PHP values with explicit contracts

Anonymous · 26/10/2026

Value Validation and Conversion

github derafu/value last commit yesterday CI passing code size 108.5 KiB open issues 0

downloads 62 downloads 62 this month

derafu/value provides composable types, strict scalar reads, explicit conversions, automatic interpretation and value inspection. It requires PHP 8.5 or later and derafu/translation.

Installation

```
composer require derafu/value
```

Components

All four public utility classes live in `Derafu\Value`.

Class	Responsibility	Guide
<code>ValueCaster</code>	Read scalar values strictly or request explicit conversions.	Casting
<code>ValueParser</code>	Interpret individual values, with optional numeric format and precision preservation.	Parsing
<code>ValueType</code>	Compose strict checks, explicit conversions, enums, dates and collection types.	Types
<code>ValueInspector</code>	Inspect values without converting or normalizing them.	Inspection

`ValueType<T>` is a `final` `readonly` object. The other classes expose static methods. Errors support translation through [Derafu Translation](#); see [Exceptions and Translations](#).

Choose an Operation

```
use Derafu\Value\ValueCaster;  
use Derafu\Value\ValueInspector;  
use Derafu\Value\ValueParser;
```

```
use Derafu\Value\ValueType;

ValueCaster::requireInt(12);    // 12; rejects null and non-integers
ValueCaster::getInt(null);     // null; rejects other non-integers
ValueType::int()->parse(12);   // 12; strings would be rejected
ValueCaster::toInt('12.0');    // 12; the destination type is explicit
ValueParser::parse('12.0');    // 12; the parser chooses the interpretation
ValueInspector::isBlank('');   // true; inspection does not convert anything
```

Strict types validate the supplied PHP type. ValueCaster provides `getXXX()/requireXXX()` for strict scalar reads and `toXXX()` for explicit conversions. String reads additionally accept `Stringable`. The parser recognizes supported text representations and keeps unrecognized text intact. None of these operations reads array keys or assigns defaults.

Null and Collections

`ValueCaster::getXXX()` and `toXXX()` preserve null; `requireXXX()` rejects it. A conversion type such as `ValueType::toInt()` rejects it unless `nullable()` is explicitly applied. The parser returns native non-text values unchanged, including `null`, arrays, objects without `Stringable` and resources.

Collection types validate or convert their elements because that traversal is explicitly part of their type definition. `ValueParser` processes one value and does not recurse into arrays.

Last updated on 26/10/2026