Introduction

Elegant Data Structure Navigation for PHP

Anonymous · 09/09/2026

Elegant Data Structure Navigation for PHP

A powerful, flexible, and efficient PHP library for querying, extracting, and modifying complex data structures using a **declarative selector syntax**.

Why Derafu\Selector?

A Smarter Way to Work with Data Structures

Traditional PHP methods for accessing and modifying nested data structures (`$array['key']['subkey']`) are:

- **Error-prone**: Risk of `Undefined index` or `Trying to access array offset on null` errors.
- **Verbose & Hard to Read**: Requires multiple `isset()` checks and deep nesting.
- **Limited**: No built-in support for **filters, conditional selectors, or transformations**.

What Makes Derafu\Selector Unique?

Feature	Derafu\Selector	PHP Native (`isset()` & loops)	Other Libraries
Dot-Notation Access	Yes	No	⚠ Limited
Nested Selection with Conditions	Yes	No	⚠ Limited
Dynamic Path Resolution	Yes	No	⚠ Varies
Auto-Handles Undefined Keys	Yes	No	⚠ Partial
Filters & Expressions	Yes	No	⚠ Partial

Overview

Derafu Selector gives you a clean, robust way to work with nested arrays and objects in PHP. It eliminates the need for repetitive null checks, nested loops, and error-prone array access, replacing them with a powerful, declarative syntax.

```
// Instead of this:
$userName = isset($data['user']) && isset($data['user']['profile']) &&
isset($data['user']['profile']['name'])
    ? $data['user']['profile']['name']
    : null;

// Simply write this:
$userName = Selector::get($data, 'user.profile.name');

// Or even filter arrays with conditions:
$adminEmail = Selector::get($data, 'users[role=admin:email]');
```

Features

- **Powerful Dot Notation** - Access deeply nested data with simple paths (`user.profile.name`).
- **Array Access** - Use numeric indices to access array elements (`items[0].name`).
- **Filtering** - Find array elements by key/value conditions (`users[id=123:email]`).
- **Conditional Logic** - Use ternary-like expressions (`((type) = "admin" ? (admin_role) : (user_role))`).
- **Multiple Query Languages** - Support for native syntax, JSONPath, and JMESPath.
- **Error Handling** - No more “undefined index” or “trying to access array offset on null” errors.
- **Write Support** - Modify data structures with the same powerful syntax.
- **Zero Dependencies** - Lightweight core with optional support for JSONPath and JMESPath.

Installation

```
composer require derafu/selector
```

Quick Start

Reading Data

```
use Derafu\Selector\Selector;
```

```
$data = [
    'user' => [
        'name' => 'John Doe',
        'email' => 'john@example.com',
        'profile' => [
            'avatar' => 'john.jpg',
            'settings' => ['theme' => 'dark', 'notifications' => true],
        ],
        'roles' => ['editor', 'contributor'],
    ],
    'products' => [
        ['id' => 101, 'name' => 'Laptop', 'price' => 999],
        ['id' => 102, 'name' => 'Phone', 'price' => 699],
        ['id' => 103, 'name' => 'Headphones', 'price' => 149],
    ],
];

// Simple property access.
$name = Selector::get($data, 'user.name');  // "John Doe"

// Nested properties.
$theme = Selector::get($data, 'user.profile.settings.theme');  // "dark"

// Array elements by index.
$firstRole = Selector::get($data, 'user.roles[0]');  // "editor"

// Array filtering.
$laptop = Selector::get($data, 'products[id=101:name]');  // "Laptop"

// With default values.
$address = Selector::get($data, 'user.address', 'Not specified');  // "Not specified"

// Check if a path exists.
$hasAvatar = Selector::has($data, 'user.profile.avatar');  // true
```

Writing Data

```
// Set a simple value.
Selector::set($data, 'user.address', '123 Main St');

// Create nested structures automatically.
Selector::set($data, 'user.profile.settings.language', 'en');

// Update array elements.
Selector::set($data, 'products[id=101:price]', 899);

// Remove a value.
Selector::clear($data, 'user.profile.settings.notifications');
```

Advanced Selector Syntax

```
// Conditional selectors.
$role = Selector::get($data, '((user.type) = "admin" ? (admin_role) : (user_role))');

// String concatenation.
$greeting = Selector::get($data, '"Hello, "(user.name)"!"');  // "Hello, John Doe!"

// OR operator for fallbacks.
$contactInfo = Selector::get($data, 'user.phone||user.email');  // Uses email if phone
is null

// JSONPath integration.
$expensiveProducts = Selector::get($data, '$.products[?(@.price > 500)].name');  //
["Laptop", "Phone"]

// JMESPath integration.
$productNames = Selector::get($data, 'jmespath:products[*].name');  // ["Laptop",
"Phone", "Headphones"]
```

Last updated on 09/09/2026