



- [Downloads](#)
- [Documentation](#)
- [Get Involved](#)
- [Help](#)
- 

[Dutch PHP Conference 2024](#)

[Getting Started](#)

[Introduction](#)

[A simple tutorial](#)

[Language Reference](#)

[Basic syntax](#)

[Types](#)

[Variables](#)

[Constants](#)

[Expressions](#)

[Operators](#)

[Control Structures](#)

[Functions](#)

[Classes and Objects](#)

[Namespaces](#)

[Enumerations](#)

[Errors](#)

[Exceptions](#)

[Fibers](#)

[Generators](#)

[Attributes](#)

[References Explained](#)

[Predefined Variables](#)

[Predefined Exceptions](#)

[Predefined Interfaces and Classes](#)

[Predefined Attributes](#)

[Context options and parameters](#)

[Supported Protocols and Wrappers](#)

[Security](#)

[Introduction](#)

[General considerations](#)

[Installed as CGI binary](#)

[Installed as an Apache module](#)

[Session Security](#)

[Filesystem Security](#)

[Database Security](#)

[Error Reporting](#)

[User Submitted Data](#)

[Hiding PHP](#)

[Keeping Current](#)

[Features](#)

[HTTP authentication with PHP](#)

[Cookies](#)

[Sessions](#)

[Dealing with XForms](#)

[Handling file uploads](#)

[Using remote files](#)

[Connection handling](#)

[Persistent Database Connections](#)

[Command line usage](#)

[Garbage Collection](#)
[DTrace Dynamic Tracing](#)

[Function Reference](#)

[Affecting PHP's Behaviour](#)
[Audio Formats Manipulation](#)
[Authentication Services](#)
[Command Line Specific Extensions](#)
[Compression and Archive Extensions](#)
[Cryptography Extensions](#)
[Database Extensions](#)
[Date and Time Related Extensions](#)
[File System Related Extensions](#)
[Human Language and Character Encoding Support](#)
[Image Processing and Generation](#)
[Mail Related Extensions](#)
[Mathematical Extensions](#)
[Non-Text MIME Output](#)
[Process Control Extensions](#)
[Other Basic Extensions](#)
[Other Services](#)
[Search Engine Extensions](#)
[Server Specific Extensions](#)
[Session Extensions](#)
[Text Processing](#)
[Variable and Type Related Extensions](#)
[Web Services](#)
[Windows Only Extensions](#)
[XML Manipulation](#)
[GUI Extensions](#)

Keyboard Shortcuts

?	This help
j	Next menu item
k	Previous menu item
g p	Previous man page
g n	Next man page
G	Scroll to bottom
g g	Scroll to top
g h	Goto homepage
g s	Goto search (current page)
/	Focus search box

[Константы перечислений »](#)
[« Методы перечислений](#)

- [Руководство по PHP](#)
- [Справочник языка](#)
- [Перечисления](#)

Change language: Russian

Статические методы перечислений

В перечислениях разрешено объявлять статические методы. Статические методы в самом перечислении в первую очередь выступают в роли альтернативных конструкторов. Например:

```
<?php

enum Size
{
    case Small;
    case Medium;
    case Large;

    public static function fromLength(int $cm): static
    {
        return match(true) {
            $cm < 50 => static::Small,
            $cm < 100 => static::Medium,
            default => static::Large,
        };
    }
}
```

Статические методы разрешено объявлять общедоступными, закрытыми или защищёнными, хотя на практике закрытые и защищённые методы эквивалентны, поскольку наследование не разрешено.

[+add a note](#)

User Contributed Notes 5 notes

[up](#)

[down](#)

53

[niloofarfs](#)

1 year ago

To get all scalar equivalent values of Backed Enum as an array you could define a method in your Enum:

```
<?php

enum Suit: string
{
    case Hearts = 'H';
    case Diamonds = 'D';
    case Clubs = 'C';
    case Spades = 'S';

    public static function values(): array
    {
        return array_column(self::cases(), 'value');
    }
}

?>
```

[up](#)

[down](#)

3

[joe502357217 at qq dot com](#)

4 months ago

You simply need to use the following code as a replacement for the example provided by Aaron Saray.

This piece of code is more concise.

```
<?php
enum Suit: string
{
    case Hearts = 'H';
    case Diamonds = 'D';
    case Clubs = 'C';
    case Spades = 'S';

    public static function forSelect(): array
    {
        return array_column(self::cases(), 'name', 'value');
    }
}
```

```
var_dump(Suit::forSelect());
?>
```

[up](#)

[down](#)

8

[Aaron Saray ¶](#)

1 year ago

Need to retrieve all the names and values immediately from a backed enum (for something like a select box) and you don't want to loop over `Enum::cases()`? Try this:

```
<?php
enum Suit: string
{
    case Hearts = 'H';
    case Diamonds = 'D';
    case Clubs = 'C';
    case Spades = 'S';

    public static function forSelect(): array
    {
        return array_combine(
            array_column(self::cases(), 'value'),
            array_column(self::cases(), 'name')
        );
    }
}
```

```
Suit::forSelect();
?>
```

Put `forSelect()` in a trait and use it in any enum you have that needs this functionality.

[up](#)

[down](#)

1

[lokashafeek7755 at gmail dot com ¶](#)

4 months ago

If you want to supplement the key-value pairs with additional descriptions for each enum value in the forSelect method, you can modify the array structure to include an associative array for each enum value. Here's an example:

```
<?php

enum Gender:int
{
    case Male = 1;
    case Female = 2;
```

```

public static function forSelect(): array
{
    return [
        self::Male->value => [
            'label' => 'Male',
            'description' => 'This is the Male gender',
        ],
        self::Female->value => [
            'label' => 'Female',
            'description' => 'This is the Female gender',
        ],
    ];
}

?>

```

In this updated example, each enum value is represented as an associative array with two keys: 'label' and 'description'. You can customize the label and description for each enum value accordingly.

Please note that to access the label and description for a specific enum value, you would need to use the corresponding array keys. For example, to get the label for the Male gender, you can use `self::forSelect()[self::Male->value]['label']`.

[up](#)
[down](#)

1

[Ulf](#)

4 months ago

Note, that enums are internally declared as final and thus, cannot extend each other (though, they are allowed to extend other classes).

That also means, that the `"Size::fromLength()"` method from this page's example redundantly uses `"static::"` (because there's no late static binding required), and could easily use `"self::"` or `"Size::"` instead.

See: <https://php.watch/versions/8.1/enums#enum-inheritance>

[+add a note](#)

- [Перечисления](#)
 - [Обзор перечислений](#)
 - [Основы перечислений](#)
 - [Типизированные перечисления](#)
 - [Методы перечислений](#)
 - [Статические методы перечислений](#)
 - [Константы перечислений](#)
 - [Трейты](#)
 - [Значения перечисления в постоянных выражениях](#)
 - [Отличия от объектов](#)
 - [Список значений](#)
 - [Сериализация](#)
 - [Почему перечисления не расширяемы](#)
 - [Примеры](#)
- [Copyright © 2001-2024 The PHP Group](#)
- [My PHP.net](#)
- [Contact](#)
- [Other PHP.net sites](#)
- [Privacy policy](#)

