



- [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

[Ошибки »](#)

[« Почему перечисления не расширяемы](#)

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

Change language: Russian

Примеры

Пример #1 Базовые ограниченные значения

```
<?php

enum SortOrder
{
    case Asc;
    case Desc;
}

function query($fields, $filter, SortOrder $order = SortOrder::Asc)
{
    /* ... */
}

?>
```

Функция query() теперь может безопасно работать, зная, что параметр \$order гарантированно будет либо вариантом SortOrder::Asc, либо вариантом SortOrder::Desc. Любое другое значение привело бы к исключению [TypeError](#), поэтому проверка ошибок или тестирование не нужны.

Пример #2 Расширенные эксклюзивные значения

```
<?php

enum UserStatus: string
{
    case Pending = 'P';
    case Active = 'A';
    case Suspended = 'S';
    case CanceledByUser = 'C';
}

public function label(): string
{
    return match($this) {
        static::Pending => 'В ожидании',
        static::Active => 'Активный',
        static::Suspended => 'Приостановленный',
        static::CanceledByUser => 'Отменено пользователем',
    };
}

?>
```

В этом примере статус пользователя может быть исключительно одним из следующих вариантов: UserStatus::Pending, UserStatus::Active, UserStatus::Suspended или UserStatus::CanceledByUser. Функция может ввести параметр UserStatus и затем принять только эти четыре значения, точка.

У всех четырёх значений есть метод label(), который возвращает удобочитаемую строку. Эта строка не зависит от скалярной эквивалентной строки "machine name", которую можно использовать, например, в поле базы данных или значении выпадающего списка HTML.

```
<?php

foreach (UserStatus::cases() as $case) {
    printf('<option value="%s">%s</option>\n', $case->value, $case->label());
}

?>
```

User Contributed Notes 1 note

[up](#)

[down](#)

2

[php-net at mentordosnerds dot com ¶](#)

5 months ago

Additional use-case examples:

<?php

```
trait EnumNamesTrait
{
    abstract public static function cases(): array;

    public static function names(): array
    {
        return array_map(fn($enum) => $enum->name, static::cases());
    }
}

trait EnumValuesTrait
{
    abstract public static function cases(): array;

    public static function values(): array
    {
        return array_map(fn($enum) => $enum->value, static::cases());
    }
}

trait EnumArraySerializableTrait
{
    use EnumNamesTrait;
    use EnumValuesTrait;

    public static function array(): array
    {
        return array_combine(static::names(), static::values());
    }
}

trait EnumJsonSerializableTrait
{
    use EnumArraySerializableTrait;

    public static function jsonSerialize(): string
    {
        return json_encode(static::array());
    }
}

enum Suit: string
{
    case Clubs = '♣';
    case Diamonds = '♦';
    case Hearts = '♥';
    case Spades = '♠';

    use EnumJsonSerializableTrait;

    public const DEFAULT = self::Hearts;
```

```
public static function default(): static
{
return self::DEFAULT;
}
}
```

```
var_dump(
Suit::cases(),
Suit::values(),
Suit::names(),
Suit::array(),
Suit::jsonSerialize(),
Suit::default(),
);
```

[+ 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](#)

