PHP 8.1
Fitur Baru

Eko Kurniawan Khannedy

Eko Kurniawan Khannedy

- Technical architect at one of the biggest ecommerce company in Indonesia
- 10+ years experiences
- www.programmerzamannow.com
- youtube.com/c/ProgrammerZamanNow

Eko Kurniawan Khannedy

- Telegram : @khannedy
- Facebook : fb.com/ProgrammerZamanNow
- Instagram : instagram.com/programmerzamannow
- Youtube : youtube.com/c/ProgrammerZamanNow
- Telegram Channel : t.me/ProgrammerZamanNow
- Email : echo.khannedy@gmail.com

Sebelum Belajar

- PHP Dasar
- PHP OOP
- PHP 8

Agenda

- Enumerations
- Readonly Properties
- First-class Callable Syntax
- New in initializers
- Pure Intersection Types
- Never return type
- Final class constants
- Fibers

Enumerations

Enumerations

- Salah satu fitur yang di bahasa pemrograman lain sudah ada, namun akhirnya baru ada di PHP 8.1 adalah enum
- Enum adalah tipe data yang nilainya terbatas
- Sebelum ada enum, biasanya kita manual membuat constant di PHP
- Contoh, misal ketika kita akan membuat tipe data Gender, Level, dan sejenisnya
- Untuk membuat enum, kita bisa menggunakan kata kunci enum

Kode : Enumerations

```php
enum Gender
{
    case Male;
    case Female;
}

class Customer
{
    public function __construct(public string $name, public Gender $gender)
    {

    }
```

Menggunakan Enum

- Untuk menggunakan enum, kita bisa gunakan seperti mengakses static field
- Dan Enum memiliki static function cases() yang mengembalikan semua array enum nya

Kode : Menggunakan Enum

```php
function sayHello(Customer $customer): string
{
    if ($customer->gender == Gender::Male) {
        return "Hello Mr.$customer->name";
    } else if ($customer->gender == Gender::Female) {
        return "Hello Mrs.$customer->name";
    } else {
        return "Hello $customer->name";
    }
}

var_dump(sayHello(new Customer("Budi", Gender::Male)));
var_dump(sayHello(new Customer("Sarah", Gender::Female)));
```

Backed Enumerations

- Kadang, saat membuat enum, kita ingin data enum tersebut merepresentasikan data lainnya, misal data string atau integer misalnya
- PHP enum mendukung hal tersebut, caranya kita bisa menambahkan titik dua, lalu diikuti dengan tipe data nya
- Misal enum Gender : string, artinya enum Gender ini bisa merepresentasikan tipe data string

Kode : Backed Enumerations

```
enum Gender: string
{
    case Male = "Mr";
    case Female = "Mrs";
}
```

Konversi Backed Enumerations

- Salah satu keuntungan menggunakan backed enumerations adalah, kita bisa melakukan konversi tipe data enum nya ke tipe data backed nya
- Kita bisa gunakan static function from() atau tryFrom() untuk konversi ke enum
- Atau menggunakan field value untuk mendapatkan tipe data backed nya

Kode : Konversi Backed Enumerations

```php
function sayHello(Customer $customer): string
{
    if ($customer->gender == null) {
        return "Hello $customer->name";
    } else {
        return "Hello " . $customer->gender->value . "." . $customer->name;
    }
}

var_dump(sayHello(new Customer("Budi", Gender::from("Mr"))));
var_dump(sayHello(new Customer("Sarah", Gender::from("Mrs"))));
```

Enumerations Method

- Enum di PHP mirip seperti class biasanya, dia bisa memiliki method atau function

Kode : Enumerations Method

```php
enum Gender: string
{
    case Male = "Mr";
    case Female = "Mrs";

    function sayHello(): string
    {
        return "Hello " . $this->value;
    }

    function inIndonesia(): string
    {
        return match ($this) {
            Gender::Male => "Tuan",
            Gender::Female => "Nyonya"
        };
    }
```

Enumerations Static Method

- Selain method, kita juga bisa menambahkan static method di enum

Kode : Enumerations Static Method

```php
enum Gender: string
{
    case Male = "Mr";
    case Female = "Mrs";

    static function fromIndonesia(string $value): Gender
    {
        return match ($value) {
            "Tuan" => Gender::Male,
            "Nyonya" => Gender::Female,
            default => throw new Exception("Unsupported value")
        };
    }
```

Enumerations Inheritance

- Enun juga bisa melakukan pewarisan di PHP, namun terbatas hanya dengan interface dan trait, tidak bisa dengan class atau enum lainnya

Kode : Enumerations Interface

```php
interface SayHello
{
    function sayHello(): string;
}

enum Gender: string implements SayHello
{
    case Male = "Mr";
    case Female = "Mrs";

    function sayHello(): string
    {
        return "Hello " . $this->value;
    }
```

Kode : Enumerations Trait

```php
trait IndonesiaGender
{
    function inIndonesia(): string
    {
        return match ($this) {
            Gender::Male => "Tuan",
            Gender::Female => "Nyonya"
        };
    }
}

enum Gender: string implements SayHello
{
    use IndonesiaGender;
```

Enumerations Constant

- Enum juga bisa memiliki constant
- Tapi enum tidak bisa memiliki property/attribute

Kode : Enumerations Constant

```php
enum Gender: string implements SayHello
{
    use IndonesiaGender;

    case Male = "Mr";
    case Female = "Mrs";

    const Unknown = "Unknown";
```

Readonly Properties

Readonly Properties

- Di PHP 8.1 sekarang terdapat kata kunci readonly, yang bisa digunakan di property
- Dengan menggunakan kata kunci readonly, sekarang kita bisa membuat sebuah property hanya bisa dibaca, tidak bisa diubah lagi
- Readonly properties hanya di tentukan datanya sekali, dan tidak bisa diubah lagi

Kode : Readonly Properties

```php
class Category
{
    public readonly string $id;
    public readonly string $name;

    public function __construct(string $id, string $name)
    {
        $this->id = $id;
        $this->name = $name;
    }
```

Promoted Property

- Readonly juga bisa dibuat pada fitur PHP 8 promoted property

Kode : Promoted Readonly Property

```php
class Category
{
    public function __construct(public readonly string $id,
                                public readonly string $name)
    {

    }
```

First-class Callable Syntax

First-Class Callable Syntax

- Di PHP 8.1, sekarang kita bisa membuat sebuah reference ke function secara mudah hanya dengan menggunakan tanda ... (titik tiga kali)

Kode : Class Person

```php
class Person
{

    public function __construct(public string $name)
    {
    }

    public function sayHello(string $name): string
    {
        return "Hello $name, my name is $this->name";
    }
}
```

Kode : First-Class Callable Syntax

```php
$person = new Person("Eko");

$reference = $person->sayHello(...);

var_dump($reference("Budi"));
```

New in Initializers

New in Initializers

- Sekarang kita bisa menggunakan default value di parameter dengan new initializer
- Sebelumnya, kita tidak bisa menggunakan kata kunci new ketika membuat default value

Kode : New in Constructor Parameter

```php
require_once __DIR__ . '/Category.php';

class Product
{

    public function __construct(public string   $name,
                                public Category $category = new Category("0", "Unknown"))
    {
    }
}
```

Kode : New in Function Parameter

```php
function printProduct(Product $product = new Product("Unknown"))
{
    echo $product->name . " " . $product->category->name . PHP_EOL;
}

printProduct();
printProduct(new Product("Laptop"));
```

Pure Intersection Types

Pure Intersection Types

- Di PHP 8, diperkenalkan dengan Union Types, dimana kita bisa membuat properties atau parameter yang bisa digunakan untuk beberapa tipe data
- Di PHP 8.1 terdapat Intersection Types, dimana kita bisa membuat properties atau parameter yang wajib sesuai dengan beberapa tipe data
- Ini cocok ketika kita ingin menentukan misal parameter harus merupakan tipe data turunan beberapa interface misalnya
- Jika Union Types menggunakan karakter |, sedangkan Intersection Types menggunakan karaketer &
- Sama seperti Union Types, untuk Intersection Types juga bisa digunakan untuk beberapa tipe data, tanpa ada batasan

Kode : Contoh Interface

```php
interface HasBrand
{
    function getBrand(): string;
}

interface HasName
{
    function getName(): string;
}
```

Kode : Intersection Types

```php
function printBrandAndName(HasBrand & HasName $value)
{
    echo $value->getBrand() . "-" . $value->getName() . PHP_EOL;
}
```

Never Return Type

Never Return Type

- Sebelumnya kita tahu ada kata kunci void untuk digunakan pada return value, untuk menandai bahwa function tersebut tidak mengembalikan value apapun
- Di PHP 8.1 terdapat return type baru, yaitu never
- Never merupakan penanda bahwa sebuah function tidak akan mengembalikan value dan function akan throw exception atau menghentikan eksekusi script dengan memanggil function die(), exit() atau trigger_error()
- Never ini cocok ketika kita ingin membuat function yang setelah itu kita ingin menghentikan eksekusi kode program php nya misal

Kode : Never

```php
function stop(): never
{
    echo "Stop" . PHP_EOL;
    exit();
}

stop();

echo "Hello World" . PHP_EOL;
```

Final Class Constants

Final Class Constant

- Sebelumnya saat membuat constant di Parent Class, kita bisa override di Child Class nya
- Di PHP 8.1 sekarang kita bisa menambahkan kata kunci final di constant nya, sehingga tidak bisa di override di class child nya

Kode : Final Class Constant

```
class Foo
{
    final const XX = "Foo";
}

class Bar extends Foo
{
    const XX = "Bar"; // error
}
```

Fitur Baru Lainnya

Fitur PHP 8.1 Lainnya

- https://www.php.net/releases/8.1/en.php