Generators in PHP

•••

github.com/SonOfHarris/laravel-meetup-generators

A **simple** way of creating a **forward-only iterator**

...

Gives you the next item in a foreach loop
Potential benefits in efficient processing and memory usage
Infinite data sets
Since PHP 5.5+

Creating a generator

```
yield $value;
```

When a **yield** keyword is used in a function, the function then acts as a generator.

The function can contain any number of **yield** keywords.

The function no longer needs a **return** keyword.

```
php gen run-example array-return
php gen run-example yield
php gen generate-data --length=10
```

Keys

```
yield $key => $value;
```

The **yield** keyword can also specify the **key** with the **value**.

If the key is not provided, the generator will act like an **indexed array**, where the key starts at zero and increases sequentially.

Note that as it's an iterator and not an array, there can be **duplicate keys**.

```
php gen run-example yield-indexed
php gen run-example yield-assoc
```

Memory Usage

Arrays vs Generators

Generators are a **good fit for processing data**, particularly translation or transformation, especially where they are used **throughout the chain**.

Note that source data still needs to be stored somewhere.

```
php gen bench Data/BasicImport
php gen bench Data/MapImport
```

Processing

Arrays vs Generators

The first item is returned to the caller as soon as the **yield** keyword is used.

Processing can then only be performed **as needed**.

Added benefits when coupled with **asynchronous calls** to other systems (e.g. APIs).

```
php gen bench Data/TimeToFirst
php gen bench Data/TimeToLast
```

Flattening

```
yield from $source;
```

The **yield from** statement simplifies yielding values from **any iterable**.

Allows **multiple iterables** to be looped over as if they are **one iterable**.

Useful for seamlessly handling **batched processes**.

There is the potential for **duplicate keys** as they are preserved.

```
php gen run-example user-api
php gen bench Data/UserApi
```

Type Hints

What was it again?
An array or a generator?

Jury is still out on a standard.

Return type as **Generator** class, not **iterable** as that includes arrays.

PHPDoc **@return** tag to **type hint contents** of generator.

PHPStan has a more complex notation.

---

src/Data/UserCollection.php

Not an array

But it is an iterable

Can not be used when function expects an **array**.

Function could ask for an **iterable** type instead.

The **array_*** functions can not handle generators.

Convert using **iterator_to_array** before call.

Type hinting should allow static analysis tools to detect where there is a **type mismatch**.

```
php gen run-example array-functions
tests/Data/MapImportBench
```

Destination unknown

Are we there yet?

The **count()** function can not support generators.

Create a class that implements the **Countable interface** and has an **__invoke** method that is the generator.

src/Data/UserDataSource.php

Interrupts

Leaving it hanging

Interrupting the loop causes the code after the yield to **not be run**.

Use **try { .. } finally { ... }** blocks to ensure the code in the **finally block is run**.

```
php gen run-example resources
```

Laravel

Examples in the framework

Lazy Collections

Eloquent (late hydration)

- Lazy: supports eager loading of relationships
- Cursor: does not support eager loading of relationships

Do you **really** need an array if you're just going to loop over it?

...

Any questions?

github.com/SonOfHarris

phpaustralia.slack.com