



# Laravel Collection

Eko Kurniawan Khannedy

# Eko Kurniawan Khannedy

- Technical architect at one of the biggest ecommerce company in Indonesia
- 12+ years experiences
- [www.programmerzamannow.com](http://www.programmerzamannow.com)
- [youtube.com/c/ProgrammerZamanNow](https://youtube.com/c/ProgrammerZamanNow)





# Eko Kurniawan Khannedy

- Telegram : [@khannedy](https://t.me/khannedy)
- LinkedIn : <https://www.linkedin.com/company/programmer-zaman-now/>
- Facebook : [fb.com/ProgrammerZamanNow](https://fb.com/ProgrammerZamanNow)
- Instagram : [instagram.com/programmerzamannow](https://instagram.com/programmerzamannow)
- Youtube : [youtube.com/c/ProgrammerZamanNow](https://youtube.com/c/ProgrammerZamanNow)
- Telegram Channel : [t.me/ProgrammerZamanNow](https://t.me/ProgrammerZamanNow)
- Tiktok : <https://tiktok.com/@programmerzamannow>
- Email : echo.khannedy@gmail.com



# Sebelum Belajar

- Sudah mengikuti kelas PHP dari Programmer Zaman Now
- Sudah mengikuti kelas Laravel Dasar



# Agenda

- Pengenalan Laravel Collection
- Collection Method
- Lazy Collection
- Dan lain-lain

---

# Laravel Collection



# Masalah Array di PHP

- Saat kita menggunakan PHP, kita sering sekali menggunakan data array
- Namun array hanyalah tempat untuk menyimpan data, ketika kita butuh melakukan operasi terhadap data di array nya, maka kita harus lakukan secara manual
- Hal ini kadang menyulitkan ketika operasi yang perlu kita lakukan sangat kompleks
- Di bahasa pemrograman seperti Java, terdapat tipe data Collection untuk mempermudah hal tersebut, sayangnya hal ini tidak ada di PHP



# Laravel Collection

- Laravel menyediakan tipe data Collection untuk menangani keterbatasan tipe data array
- Collection sendiri sebenarnya seperti array, hanya dia direpresentasikan dalam class, dan memiliki banyak sekali method/function yang bisa digunakan untuk mempermudah ketika kita butuh melakukan manipulasi data Collection tersebut
- <https://laravel.com/api/10.x/Illuminate/Support/Collection.html>



---

# Membuat Project



# Membuat Project

```
composer create-project laravel/laravel=v10.0.3 belajar-laravel-collection
```

---

# Membuat Collection



# Membuat Collection

- Untuk membuat Collection, Laravel sudah menyediakan global function `collect(array)` yang digunakan untuk mengubah tipe data array menjadi Collection

## Kode : Membuat Collection

```
10 ✓ class CollectionTest extends TestCase
11 {
12 ✓ public function testCreateCollection()
13 {
14     $collection = collect([1, 2, 3]);
15     $this->assertEquals([1, 2, 3], $collection->all());
16 }
17
18 }
```

---

# For Each



# For Each

- Collection adalah struktur data turunan dari Iterable PHP
- Dengan demikian, kita bisa melakukan iterasi data Collection menggunakan perintah for PHP



## Kode : For Each

```
public function testForEach()
{
    $collection = collect([1, 2, 3, 4, 5, 6, 7, 8, 9]);
    foreach ($collection as $key => $value) {
        self::assertEquals( expected: $key + 1, $value);
    }
}
```



---

# Manipulasi Collection



# Manipulasi Collection

- Collection adalah sebuah class, oleh karena itu untuk memanipulasi data nya, kita perlu menggunakan method yang terdapat di Collection



# Collection Operations

Method	Keterangan
push(data)	Menambah data ke paling belakang
pop()	Menghapus dan mengambil data paling terakhir
prepend(data)	Menambah data ke paling depan
pull(key)	Menghapus dan mengambil data sesuai dengan key
put(key, data)	Mengubah data dengan key



## Kode : Manipulasi Collection

```
public function testCrud()
{
    $collection = collect([]);
    $collection->push(...values: 1, 2, 3);
    assertEqualsCanonicalizing([1, 2, 3], $collection->all());

    $result = $collection->pop();
    assertEquals(expected: 3, $result);
    assertEqualsCanonicalizing([1, 2], $collection->all());
}
```

---

# Mapping



# Mapping

- Mapping adalah transformasi (mengubah bentuk data) menjadi data lain
- Mapping membutuhkan function sebagai parameter yang digunakan untuk membentuk data lainnya
- Urutan Collection hasil mapping adalah sama dengan urutan collection aslinya



# Mapping Operations

Method	Keterangan
map(function)	Iterasi seluruh data, dan mengirim seluruh data ke function
mapInto(class)	Iterasi seluruh data, dan membuat object baru untuk class dengan mengirim parameter tiap data
mapSpread(function)	Iterasi seluruh data, dan mengirim tiap data sebagai parameter di function
mapToGroups(function)	Iterasi seluruh data, dan mengirim tiap data ke function, function harus mengembalikan single key-value array untuk di group sebagai collection baru



## Kode : Map

```
public function testMap()
{
    $collection = collect([1, 2, 3]);
    $result = $collection->map(function ($item) {
        return $item * 2;
    });
    $this->assertEquals([2, 4, 6], $result->all());
}
```





## Kode : Map Into (1)

```
2 namespace App\Data;
```

```
4 no usages
```

```
5 class Person
```

```
6 {
```

```
7     var string $name;
```

```
8 no usages
```

```
9 public function __construct(string $name)
```

```
10 {
```

```
11     $this->name = $name;
```

```
12 }
```

```
13 }
```

```
14 |
```



## Kode : Map Into (2)

```
public function testMapInto()  
{  
    $collection = collect(["Eko"]);  
    $result = $collection->mapInto( class: Person::class);  
    $this->assertEquals([new Person( name: "Eko")], $result->all());  
}
```



## Kode : Map Spread

```
public function testMapSpread()
{
    $collection = collect([["Eko", "Kurniawan"], ["Khannedy", "Setiawan"]]);
    $result = $collection->mapSpread(function ($firstName, $lastName) {
        $fullName = $firstName . " " . $lastName;
        return new Person($fullName);
    });
    assertEquals([
        new Person( name: "Eko Kurniawan"),
        new Person( name: "Khannedy Setiawan")
    ], $result->all());
}
```



## Kode : Map To Groups

```
$collection = collect([
  [
    "name" => "Eko",
    "department" => "IT"
  ],
  [
    "name" => "Khannedy",
    "department" => "IT"
  ],
  [
    "name" => "Budi",
    "department" => "HR"
  ]
]);
```

```
$result = $collection->mapToGroups(function ($item) {
    return [$item["department"] => $item["name"]];
});
assertEquals([
    "IT" => collect(["Eko", "Khannedy"]),
    "HR" => collect(["Budi"])
], $result->all());
}
```

---

# Ziping



# Zippping

- Zippping adalah transformasi yang digunakan untuk menggabungkan dua buah collection.



# Zipping Operations

Method	Keterangan
zip(collection/array)	Menggabungkan tiap item di collection sehingga menjadi collection baru
concat(collection/array)	Menambahkan collection pada bagian akhir sehingga menjadi collection baru
combine(collection/array)	Menggabungkan collection sehingga collection pertama menjadi key dan collection kedua menjadi value



## Kode : Zip

```
public function testZip()
{
    $collection1 = collect([1, 2, 3]);
    $collection2 = collect([4, 5, 6]);
    $collection3 = $collection1->zip($collection2);

    assertEquals([
        collect([1, 4]),
        collect([2, 5]),
        collect([3, 6])
    ], $collection3->all());
}
```





## Kode : Concat

```
public function testConcat()
{
    $collection1 = collect([1, 2, 3]);
    $collection2 = collect([4, 5, 6]);
    $collection3 = $collection1->concat($collection2);

    assertEquals([1, 2, 3, 4, 5, 6], $collection3->all());
}
```



## Kode : Combine

```
public function testCombine()
{
    $collection1 = ["name", "country"];
    $collection2 = ["Eko", "Indonesia"];
    $collection3 = collect($collection1)->combine($collection2);

    assertEquals([
        "name" => "Eko",
        "country" => "Indonesia"
    ], $collection3->all());
}
```

---

# Flattening



# Flattening

- Flattening adalah operasi transformasi untuk menjadikan nested collection menjadi flat



# Zippping Operations

Method	Keterangan
<code>collapse()</code>	Mengubah tiap array di item collection menjadi flat collection
<code>flatMap(function)</code>	Iterasi tiap data, dikirim ke function yang menghasilkan collection, dan diubah menjadi flat collection



## Kode : Collapse

```
public function testCollapse()
{
    $collection = collect([
        [1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]
    ]);
    $result = $collection->collapse();
    assertEquals([1, 2, 3, 4, 5, 6, 7, 8, 9], $result->all());
}
```



## Kode : Flat Map

```
public function testFlatMap()
{
    $collection = collect([
        [
            "name" => "Eko",
            "hobbies" => ["Coding", "Gaming"]
        ],
        [
            "name" => "Khannedy",
            "hobbies" => ["Reading", "Writing"]
        ]
    ]);

    $hobbies = $collection->flatMap(function ($item) {
        return $item["hobbies"];
    });

    assertEquals(["Coding", "Gaming", "Reading", "Writing"], $hobbies->all());
}
```

---

# String Representation





# String Representation

- String Representations adalah operasi transformasi untuk mengubah collection menjadi string



# String Representation Operations

Method	Keterangan
<code>join(glue = "", finalGlue = "")</code>	Mengubah tiap item menjadi string dengan menggabungkan dengan separator glue, dan separator akhir finalGlue



## Kode : String Representation

```
public function testJoin()
{
    $collection = collect(["Eko", "Khannedy", "Khannedy"]);

    assertEquals( expected: "Eko-Khannedy-Khannedy", $collection->join( glue: "-" ));
    assertEquals( expected: "Eko-Khannedy_Khannedy", $collection->join( glue: "-", finalGlue: "_" ));
}
```

---

# Filtering



# Filtering

- Filtering adalah salah satu operasi collection yang banyak digunakan
- Filtering membutuhkan function sebagai parameter
- Jika function mengembalikan true, maka data akan diambil, jika false, maka data akan dibuang
- Hati-hati ketika menggunakan data array dengan index number, karena data array bisa dihapus sehingga index akan hilang di Collection baru hasil filter



# Filtering Operations

Method	Keterangan
<code>filter(function)</code>	Iterasi setiap data, dikirim ke function, jika true maka data diambil, jika false, maka data dibuang



## Kode : Filter

```
public function testFilter()
{
    $collection = collect([
        "Eko" => 100,
        "Budi" => 80,
        "Joko" => 90
    ]);
    $result = $collection->filter(function ($item, $key) {
        return $item >= 90;
    });
    assertEquals([
        "Eko" => 100,
        "Joko" => 90
    ], $result->all());
}
```

---

# Partitioning





# Partitioning

- Pada filtering, kita akan kehilangan data yang tidak masuk kondisi filter function nya
- Dengan partitioning, kita bisa mendapatkan dua collection yang terdiri dari collection yang masuk filter dan yang tidak masuk filter



# Partitioning Operations

Method	Keterangan
<code>partition(function)</code>	Iterasi setiap data, dikirim ke function, jika true maka data akan masuk ke collection pertama, jika false maka data akan masuk ke collection kedua



## Kode : Partition

```
public function testPartition()
{
    $collection = collect([
        "Eko" => 100,
        "Budi" => 80,
        "Joko" => 90
    ]);
    [$result1, $result2] = $collection->partition(function ($item, $key) {
        return $item >= 90;
    });
    assertEquals(["Eko" => 100, "Joko" => 90], $result1->all());
    assertEquals(["Budi" => 80], $result2->all());
}
```

---

# Testing



# Testing

- Testing adalah operasi untuk mengecek isi data di collection
- Hasil dari testing adalah boolean, dimana true jika sesuai kondisi, dan false jika tidak sesuai kondisi



# Testing Operations

Method	Keterangan
has(array)	Mengecek apakah collection memiliki semua key data
hasAny(array)	Mengecek apakah collection memiliki salah satu key data
contains(value)	Mengecek apakah collection memiliki data value
contains(key, value)	Mengecek apakah collection memiliki data key dengan value
contains(function)	Iterasi tiap data, mengirim ke function dan mengecek apakah salah satu data menghasilkan true



## Kode : Testing

```
public function testTesting()
{
    $collection = collect(["Eko", "Kurniawan", "Khannedy"]);
    self::assertTrue($collection->contains(key: "Eko"));
    self::assertTrue($collection->contains(function ($value, $key){
        return $value == "Eko";
    }));
}
```

---

# Grouping





# Grouping

- Grouping adalah operasi untuk meng-grup kan element-element yang ada di collection.



# Grouping Operations

Method	Keterangan
groupBy(key)	Menggabungkan data collection per key
groupBy(function)	Menggabungkan data collection per hasil function

# Kode : Grouping

```
$collection = collect([  
  [  
    "name" => "Eko",  
    "department" => "IT"  
  ],  
  [  
    "name" => "Khannedy",  
    "department" => "IT"  
  ],  
  [  
    "name" => "Budi",  
    "department" => "HR"  
  ]  
]);
```

```
$result = $collection->groupBy( groupBy: "department");  
assertEquals([  
  "IT" => collect([  
    [  
      "name" => "Eko",  
      "department" => "IT"  
    ],  
    [  
      "name" => "Khannedy",  
      "department" => "IT"  
    ]  
  ]),  
  "HR" => collect([  
    [  
      "name" => "Budi",  
      "department" => "HR"  
    ]  
  ])  
], $result->all());
```

```
assertEquals([  
  "IT" => collect([  
    [  
      "name" => "Eko",  
      "department" => "IT"  
    ],  
    [  
      "name" => "Khannedy",  
      "department" => "IT"  
    ]  
  ]),  
  "HR" => collect([  
    [  
      "name" => "Budi",  
      "department" => "HR"  
    ]  
  ])  
], $collection->groupBy(function ($value, $key){  
  return $value["department"];  
})->all());
```

---

# Slicing



# Slicing

- Slicing adalah operasi untuk mengambil sebagian data di Collection



# Slicing Operations

Method	Keterangan
<code>slice(start)</code>	Mengambil data mulai dari start sampai data terakhir
<code>slice(start, length)</code>	Mengambil data mulai dari start sepanjang length



## Kode : Slicing

```
public function testSlice()
{
    $collection = collect([1, 2, 3, 4, 5, 6, 7, 8, 9]);
    $result = $collection->slice( offset: 3);
    assertEqualsCanonicalizing([4, 5, 6, 7, 8, 9], $result->all());

    $result = $collection->slice( offset: 3, length: 2);
    assertEqualsCanonicalizing([4, 5], $result->all());
}
```

---

# Take & Skip





# Take & Skip

- Untuk mengambil sebagian element di collection, selain menggunakan slice, kita juga bisa menggunakan operator take dan skip



## Take Operations

Method	Keterangan
<code>take(length)</code>	Mengambil data dari awal sepanjang <code>length</code> , jika <code>length</code> negative artinya proses mengambil dari posisi belakang
<code>takeUntil(function)</code>	Iterasi tiap data, ambil tiap data sampai function mengembalikan nilai <code>true</code>
<code>takeWhile(function)</code>	Iterasi tiap data, ambil tiap data sampai function mengembalikan nilai <code>false</code>



## Kode : Take

```
public function testTake()
{
    $collection = collect([1, 2, 3, 4, 5, 6, 7, 8, 9]);
    $result = $collection->take(limit: 3);
    assertEqualsCanonicalizing([1, 2, 3], $result->all());

    $result = $collection->takeUntil(function ($value, $key){
        return $value == 3;
    });
    assertEqualsCanonicalizing([1, 2], $result->all());

    $result = $collection->takeWhile(function ($value, $key){
        return $value < 3;
    });
    assertEqualsCanonicalizing([1, 2], $result->all());
}
```



## Skip Operations

Method	Keterangan
<code>skip(length)</code>	Ambil seluruh data kecuali sejumlah length data diawal
<code>skipUntil(function)</code>	Iterati tiap data, jangan ambil tiap data sampai function mengembalikan nilai true
<code>skipWhile(function)</code>	Iterasi tiap data, jangan ambil tiap data sampai function mengembalikan nilai false



## Kode : Skip

```
public function testSkip()
{
    $collection = collect([1, 2, 3, 4, 5, 6, 7, 8, 9]);
    $result = $collection->skip(count: 3);
    assertEqualsCanonicalizing([4, 5, 6, 7, 8, 9], $result->all());

    $result = $collection->skipUntil(function ($value, $key){
        return $value == 3;
    });
    assertEqualsCanonicalizing([3, 4, 5, 6, 7, 8, 9], $result->all());

    $result = $collection->skipWhile(function ($value, $key){
        return $value < 3;
    });
    assertEqualsCanonicalizing([3, 4, 5, 6, 7, 8, 9], $result->all());
}
```

—

Chunked



# Chunked

- Chunked adalah operasi untuk memotong collection menjadi beberapa collection



# Chunked Operations

Method	Keterangan
<code>chunk(number)</code>	Potong collection menjadi lebih kecil dimana tiap collection memiliki sejumlah total data number





## Kode : Chunked

```
public function testChunked()
{
    $collection = collect([1, 2, 3, 4, 5, 6, 7, 8, 9]);
    $result = $collection->chunk( size: 3);

    assertEqualsCanonicalizing([1,2,3], $result->all()[0]->all());
    assertEqualsCanonicalizing([4,5,6], $result->all()[1]->all());
    assertEqualsCanonicalizing([7,8,9], $result->all()[2]->all());
}
```

---

# Retrieve



# Retrieve

- Retrieve adalah operasi untuk mengambil data di Collection



# First Operations

Method	Keterangan
<code>first()</code>	Mengambil data pertama di collection, atau null jika tidak ada
<code>firstOrNull()</code>	Mengambil data pertama di collection, atau error <code>ItemNotFoundException</code> jika tidak ada
<code>first(function)</code>	Mengambil data pertama di collection yang sesuai dengan kondisi function jika menghasilkan data true
<code>firstWhere(key, value)</code>	Mengambil data pertama di collection dimana key sama dengan value



## Kode : First

```
public function testFirst()
{
    $collection = collect([1, 2, 3, 4, 5, 6, 7, 8, 9]);
    $result = $collection->first();
    assertEquals( expected: 1, $result);

    $result = $collection->first(function ($value, $key){
        return $value > 5;
    });
    assertEquals( expected: 6, $result);
}
```



# Last Operations

Method	Keterangan
<code>last()</code>	Mengambil data terakhir di collection, atau null jika tidak ada
<code>last(function)</code>	Mengambil data terakhir di collection yang sesuai dengan kondisi function jika menghasilkan data true



## Kode : Last

```
public function testLast()
{
    $collection = collect([1, 2, 3, 4, 5, 6, 7, 8, 9]);
    $result = $collection->last();
    assertEquals( expected: 9, $result);

    $result = $collection->last(function ($value, $key){
        return $value < 5;
    });
    assertEquals( expected: 4, $result);
}
```

—

Random





# Random

- Random adalah operasi untuk mengambil data di collection dengan posisi random



# Random Operations

Method	Keterangan
random()	Mengambil satu data collection dengan posisi random
random(total)	Mengambil sejumlah total data collection dengan posisi random



## Kode : Random

```
public function testRandom()
{
    $collection = collect([1, 2, 3, 4, 5, 6, 7, 8, 9]);
    $result = $collection->random();
    self::assertTrue(in_array($result, [1, 2, 3, 4, 5, 6, 7, 8, 9]));
}
```

---

# Checking Existence



# Checking Existence

- Checking Existence merupakan operasi untuk mengecek apakah terdapat data yang dicari di Collection



## Checking Existence Operations

Method	Keterangan
isEmpty() : bool	Mengecek apakah collection kosong
isNotEmpty()	Mengecek apakah collection tidak kosong
contains(value)	Mengecek apakah collection memiliki value
contains(function)	Mengecek apakah collection memiliki value dengan kondisi function yang menghasilkan true
containsOneItem()	Mengecek apakah collection hanya memiliki satu data



## Kode : Checking Existence

```
public function testCheckingExistence()
{
    $collection = collect([1, 2, 3, 4, 5, 6, 7, 8, 9]);
    self::assertTrue($collection->isNotEmpty());
    self::assertFalse($collection->isEmpty());
    self::assertTrue($collection->contains( key: 8));
    self::assertFalse($collection->contains( key: 10));
    self::assertTrue($collection->contains(function ($value, $key){
        return $value == 8;
    }));
}
```

---

# Ordering





# Ordering

- Ordering adalah operasi untuk melakukan pengurutan data di Collection

# Ordering Operations



Method	Keterangan
sort()	Mengurutkan secara ascending
sortBy(key/function)	Mengurutkan secara ascending berdasarkan key atau function
sortDesc()	Mengurutkan secara descending
sortByDesc(key/function)	Mengurutkan secara descending berdasarkan key atau function
sortKeys()	Mengurutkan secara ascending berdasarkan keys
sortKeysDesc()	Mengurutkan secara descending berdasarkan keys
reverse()	Membalikkan urutan collection



## Kode : Ordering

```
public function testOrdering()
{
    $collection = collect([1, 3, 2, 4, 5, 7, 6, 8, 9]);
    $result = $collection->sort();
    assertEqualsCanonicalizing([1, 2, 3, 4, 5, 6, 7, 8, 9], $result->all());

    $result = $collection->sortDesc();
    assertEqualsCanonicalizing([9, 8, 7, 6, 5, 4, 3, 2, 1], $result->all());
}
```

---

# Aggregate



# Aggregate

- Laravel collection juga memiliki beberapa method untuk melakukan aggregate

# Aggregate Operations



Method	Keterangan
min()	Mengambil data paling kecil
max()	Mengambil data paling besar
avg() / average()	Mengambil rata-rata data
sum()	Mengambil seluruh jumlah data
count()	Mengambil total seluruh data



# Kode : Aggregate

```
public function testAggregate()
{
    $collection = collect([1, 2, 3, 4, 5, 6, 7, 8, 9]);
    $result = $collection->sum();
    assertEquals( expected: 45, $result);

    $result = $collection->avg();
    assertEquals( expected: 5, $result);

    $result = $collection->min();
    assertEquals( expected: 1, $result);

    $result = $collection->max();
    assertEquals( expected: 9, $result);
}
```

---

# Reduce





# Reduce

- Jika kita ingin membuat aggregate secara manual, kita bisa menggunakan function reduce
- Reduce merupakan operasi yang dilakukan pada tiap data yang ada di collection secara sequential dan mengembalikan hasil
- Hasil dari reduce sebelumnya akan digunakan di iterasi selanjutnya

# Reduce Operations



Method	Keterangan
<code>reduce(function(carry, item))</code>	<p>Pada iterasi pertama, carry akan bernilai data pertama, dan item adalah data selanjutnya</p> <p>Pada iterasi selanjutnya, carry adalah hasil dari iterasi sebelumnya</p>



## Kode : Reduce

```
public function testReduce()
{
    $collection = collect([1, 2, 3, 4, 5, 6, 7, 8, 9]);
    $result = $collection->reduce(function ($carry, $item) {
        return $carry + $item;
    });
    assertEquals( expected: 45, $result);
}
```

---

# Method Lainnya



# Metode Lainnya

- Sekarang kita sudah membahas hampir sebagian besar operasi yang bisa kita lakukan di Laravel Collection
- Namun sebenarnya masih banyak method yang bisa kita gunakan di Laravel Collection
- <https://laravel.com/api/10.x/Illuminate/Support/Collection.html>
- <https://laravel.com/docs/10.x/collections#available-methods>

---

# Lazy Collection



# Lazy Collection

- Saat belajar PHP, kita pernah membuat Generator (Lazy Array/Iterable)
- Di Laravel juga kita bisa membuat hal seperti itu, bernama Lazy Collection
- Keuntungan menggunakan Lazy Collection adalah kita bisa melakukan manipulasi data besar, tanpa harus takut semua operasi dieksekusi sebelum dibutuhkan
- Saat membuat Lazy Collection, kita perlu menggunakan PHP Generator
- <https://laravel.com/api/10.x/Illuminate/Support/LazyCollection.html>



## Kode : Lazy Collection

```
public function testLazyCollection()
{
    $collection = LazyCollection::make(function () {
        $value = 0;
        while (true) {
            yield $value;
            $value++;
        }
    });

    $result = $collection->take(limit: 10);
    assertEqualsCanonicalizing([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], $result->all());
}
```



---

# Materi Selanjutnya



# Materi Selanjutnya

- Laravel Database
- Laravel Eloquent ORM