

Les collections : la programmation fonctionnelle s'invite dans Laravel


Exemples d'utilisation

Yannick Chenot

Laravel Montréal 2019

C'est qui lui ?



- ➔ Yannick Chenot · [@osteel](#) 
- ➔ Développeur web depuis + de 10 ans
- ➔ Travaille avec Laravel depuis 3-4 ans
- ➔ Consultant chez VuduMobile
- ➔ Blogge (parfois) sur [tech.osteel.me](#)

De quoi on parle

- ➔ Les collections Laravel
- ➔ Impératif vs déclaratif
- ➔ Le jeu de données
- ➔ Un exemple simple
- ➔ Comparaison de performance
- ➔ D'autres exemples

De quoi on parle

- ➡ Utilisez les macros !
- ➡ Conclusion
- ➡ Références
- ➡ Questions

Les collections Laravel

- ➔ Surcouche des tableaux PHP
- ➔ Facilite leur utilisation en offrant une syntaxe fluide
- ➔ POO saupoudrée de programmation fonctionnelle
- ➔ Eloquent retourne des collections : `User::get()`

```
$collection = new \Illuminate\Support\Collection([1, 2, 3]);  
$collection = collect([1, 2, 3]);  
$collection->all();
```

Impératif vs déclaratif

- ➡ Impératif : dire au programme ce qu'*il doit* faire
- ➡ Déclaratif : dire au programme ce qu'*on veut* faire

Le jeu de données



Un exemple simple

- ➡ Convertir les données brutes en une liste d'objets
- ➡ Sélectionner les pokémons de type feu 🔥
- ➡ Identifier la plus forte attaque

Pokemon

Un exemple simple

```
$dataset = json_decode(file_get_contents(storage_path('pokedex.json')), true);
```

Un exemple simple

Style impératif

- ➔ Convertir les données brutes en une liste d'objets

Pokemon

```
$pokedex = [];  
foreach ($dataset as $attributes) {  
    $pokedex[] = new Pokemon($attributes);  
}
```

Un exemple simple

Style impératif

➡ Sélectionner les pokémons de type feu 🔥

```
$pokedex = [];  
foreach ($dataset as $attributes) {  
    $pokemon = new Pokemon($attributes);  
    if ($pokemon->isType('fire')) {  
        $pokedex[] = $pokemon;  
    }  
}
```

Un exemple simple

Style impératif

- ➔ Identifier la plus forte attaque

```
$attack = 0;
foreach ($dataset as $attributes) {
    $pokemon = new Pokemon($attributes);
    if (! $pokemon->isType('fire')) {
        continue;
    }
    if ($pokemon->abilities->attack > $attack) {
        $attack = $pokemon->abilities->attack;
    }
}
```

DEMO

Un exemple simple

Style déclaratif : fonctions d'ordre supérieur

- ➔ Acceptent d'autres fonctions comme arguments
- ➔ Retournent une fonction
- ➔ Fonctions de première classe, anonymes, closures
- ➔ `array_filter` `array_reduce` `array_map`

Un exemple simple

Style déclaratif : fonctions d'ordre supérieur



[Tweet](#) original

Un exemple simple

Style déclaratif : fonctions d'ordre supérieur

➡ Convertir les données brutes en une liste d'objets **Pokemon**

```
$pokedex = array_map(  
    function (array $attributes) {  
        return new Pokemon($attributes);  
    },  
    $dataset  
);
```

Un exemple simple

Style déclaratif : fonctions d'ordre supérieur

➡ Sélectionner les pokémons de type feu 🔥

```
$pokedex =  
    array_filter(  
        array_map(  
            function (array $attributes) {  
                return new Pokemon($attributes);  
            },  
            $dataset  
        ),  
        function (Pokemon $pokemon) {  
            return $pokemon->isType('fire');  
        }  
    );
```

Un exemple simple

Style déclaratif : fonctions d'ordre supérieur

- ➔ Identifier la plus forte attaque

```
$attack = array_reduce(  
    array_filter(  
        array_map(  
            function (array $attributes) {  
                return new Pokemon($attributes);  
            },  
            $dataset  
        ),  
        function (Pokemon $pokemon) {  
            return $pokemon->isType('fire');  
        }  
    ),  
    function (int $attack, Pokemon $pokemon) {  
        return $pokemon->abilities->attack > $attack ? $pokemon->abilities->attack : $attack;  
    },  
    0  
);
```



DEMO

Un exemple simple

Style déclaratif : collections Laravel



Convertir les données brutes en une liste d'objets

Pokemon

```
$pokedex = collect($dataset)
    ->map(function (array $attributes) {
        return new Pokemon($attributes);
    });
```

Un exemple simple

Style déclaratif : collections Laravel

➡ Sélectionner les pokémons de type feu 🔥

```
$pokedex = collect($dataset)
    ->map(function (array $attributes) {
        return new Pokemon($attributes);
    })
    ->filter(function (Pokemon $pokemon) {
        return $pokemon->isType('fire');
    });
```

Un exemple simple

Style déclaratif : collections Laravel

➔ Identifier la plus forte attaque

```
$attack = collect($dataset)
    ->map(function (array $attributes) {
        return new Pokemon($attributes);
    })
    ->filter(function (Pokemon $pokemon) {
        return $pokemon->isType('fire');
    })
    ->reduce(function (int $attack, Pokemon $pokemon) {
        return $pokemon->abilities->attack > $attack ? $pokemon->abilities->attack : $attack;
    }, 0);
```


Un exemple simple

Style déclaratif : fonctions d'ordre supérieur

➔ Avant...

```
$attack = array_reduce(  
    array_filter(  
        array_map(  
            function (array $attributes) {  
                return new Pokemon($attributes);  
            },  
            $dataset  
        ),  
        function (Pokemon $pokemon) {  
            return $pokemon->isType('fire');  
        }  
    ),  
    function (int $attack, Pokemon $pokemon) {  
        return $pokemon->abilities->attack > $attack ? $pokemon->abilities->attack : $attack;  
    },  
    0  
);
```

Un exemple simple

Style déclaratif : fonctions d'ordre supérieur

➡ Après

```
$attack = collect($dataset)
->map(function (array $attributes) {
    return new Pokemon($attributes);
})
->filter(function (Pokemon $pokemon) {
    return $pokemon->isType('fire');
})
->reduce(function (int $attack, Pokemon $pokemon) {
    return $pokemon->abilities->attack > $attack ? $pokemon->abilities->attack : $attack;
}, 0);
```

Un exemple simple

Style déclaratif : collections Laravel

➡ Allons plus loin...

```
$attack = collect($dataset)
    ->map(function (array $attributes) {
        return new Pokemon($attributes);
    })
    ->filter(function (Pokemon $pokemon) {
        return $pokemon->isType('fire');
    })
    ->reduce(function (int $attack, Pokemon $pokemon) {
        return $pokemon->abilities->attack > $attack ? $pokemon->abilities->attack : $attack;
    }, 0);
```

Un exemple simple

Style déclaratif : collections Laravel

➡ Allons plus loin... avec **mapInto**

```
$attack = collect($dataset)
    ->mapInto(Pokemon::class)
    ->filter(function (Pokemon $pokemon) {
        return $pokemon->isType('fire');
    })
    ->reduce(function (int $attack, Pokemon $pokemon) {
        return $pokemon->abilities->attack > $attack ? $pokemon->abilities->attack : $attack;
    }, 0);
```

Un exemple simple

Messages d'ordre supérieur

- ➔ Message qui prend un autre message comme argument
- ➔ Une méthode ou une propriété de l'objet de la collection



Un exemple simple

Messages d'ordre supérieur

➡ Soit un modèle `User` qui a une propriété `is_admin`
et une méthode `isAdmin(): bool` ...

```
User::get()->filter(function (User $user) {  
    return $user->is_admin;  
});  
  
User::get()->filter(function (User $user) {  
    return $user->isAdmin();  
});
```


Un exemple simple

Messages d'ordre supérieur

```
User::get()->filter(function (User $user) {  
    return $user->is_admin;  
});  
  
// becomes...  
User::get()->filter->is_admin;  
  
User::get()->filter(function (User $user) {  
    return $user->isAdmin();  
});  
  
// becomes...  
User::get()->filter->isAdmin();
```

Un exemple simple

Style déclaratif : collections Laravel

➡ Allons plus loin... avec les messages d'ordre supérieur

```
$attack = collect($dataset)
    ->mapInto(Pokemon::class)
    ->filter(function (Pokemon $pokemon) {
        return $pokemon->isType('fire');
    })
    ->reduce(function (int $attack, Pokemon $pokemon) {
        return $pokemon->abilities->attack > $attack ? $pokemon->abilities->attack : $attack;
    }, 0);
```

Un exemple simple

Style déclaratif : collections Laravel

➡ Allons plus loin... avec les messages d'ordre supérieur

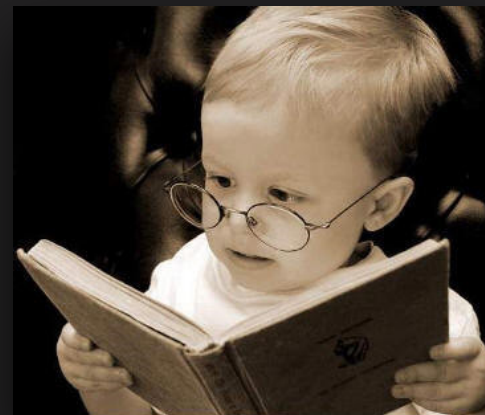
```
$attack = collect($dataset)
    ->mapInto(Pokemon::class)
    ->filter->isType('fire')
    ->reduce(function (int $attack, Pokemon $pokemon) {
        return $pokemon->abilities->attack > $attack ? $pokemon->abilities->attack : $attack;
    }, 0);
```

Un exemple simple

Style déclaratif : collections Laravel

➡ Allons plus loin... avec des méthodes plus spécialisées

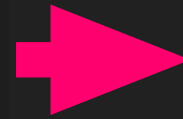
```
$attack = collect($dataset)
    ->mapInto(Pokemon::class)
    ->filter->isType('fire')
    ->map(function (Pokemon $pokemon) {
        return $pokemon->abilities->attack;
    })
    ->max();
```



Un exemple simple

Comparaison

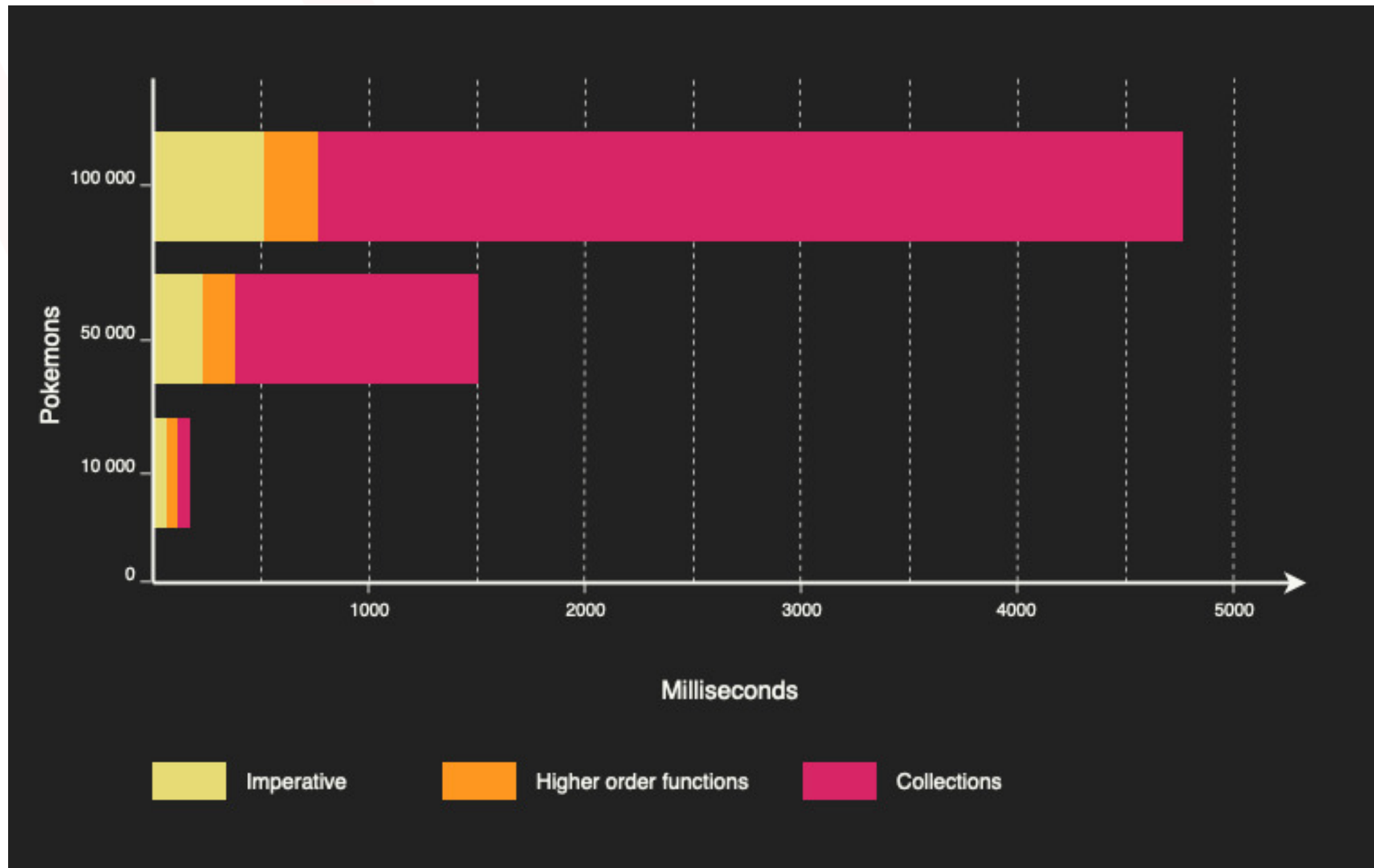
```
$attack = 0;
foreach ($dataset as $attributes) {
    $pokemon = new Pokemon($attributes);
    if (! $pokemon->isType('fire')) {
        continue;
    }
    if ($pokemon->abilities->attack > $attack) {
        $attack = $pokemon->abilities->attack;
    }
}
```



```
$attack = collect($dataset)
->mapInto(Pokemon::class)
->filter->isType('fire')
->map(function (Pokemon $pokemon) {
    return $pokemon->abilities->attack;
})
->max();
```



Comparaison de performance



D'autres exemples

```
$dataset = json_decode(file_get_contents(storage_path('pokedex.json')), true);  
$pokedex = collect($dataset)->mapInto(Pokemon::class);
```

D'autres exemples

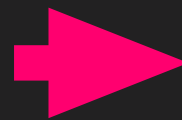
Total des aptitudes par type de pokémon

```
$abilities = $pokedex  
->groupBy('types')
```

D'autres exemples

Total des aptitudes par type de pokémon

```
[
  {
    "id": 1,
    "name": "Bulbasaur",
    "types": [
      "grass",
      "poison"
    ]
  }
]
```



```
{
  "grass": [
    {
      "id": 1,
      "name": "Bulbasaur",
      "types": [
        "grass",
        "poison"
      ]
    }
  ],
  "poison": [
    {
      "id": 1,
      "name": "Bulbasaur",
      "types": [
        "grass",
        "poison"
      ]
    }
  ]
}
```

D'autres exemples

Total des aptitudes par type de pokémon

```
$abilities = $pokedex  
->groupBy('types')
```

D'autres exemples

Total des aptitudes par type de pokémon

```
$abilities = $pokedex
->groupBy('types')
->map(function (Collection $pokemons) {
    return $pokemons->reduce(function (int $score, Pokemon $pokemon) {
        return $score + $pokemon->abilities->sum();
    }, 0);
})
```

D'autres exemples

Total des aptitudes par type de pokémon

```
$abilities = $pokedex  
  ->groupBy('types')  
  ->map->reduce(function (int $score, Pokemon $pokemon) {  
    return $score + $pokemon->abilities->sum();  
  }, 0)
```

D'autres exemples

Total des aptitudes par type de pokémon

```
$abilities = $pokedex  
  ->groupBy('types')  
  ->map->reduce(function (int $score, Pokemon $pokemon) {  
    return $score + $pokemon->abilities->sum();  
  }, 0)  
  ->sort();
```



D'autres exemples

Le pokémon le plus fort

```
$strongest = $pokedex  
->sortByDesc(function (Pokemon $pokemon) {  
    return $pokemon->abilities->sum();  
})  
->first();
```



D'autres exemples

Fuyez, pauvres fous !

```
$pokedex  
->reject(function (Pokemon $pokemon) use ($strongest) {  
    return $strongest->abilities->speed >= $pokemon->abilities->speed;  
})
```

D'autres exemples

Fuyez, pauvres fous !

```
$pokedex  
->reject(function (Pokemon $pokemon) use ($strongest) {  
    return $strongest->abilities->speed >= $pokemon->abilities->speed;  
})  
->each(function (Pokemon $pokemon) {  
    $pokemon->flee();  
});
```

D'autres exemples

Fuyez, pauvres fous !

```
$pokedex  
->reject(function (Pokemon $pokemon) use ($strongest) {  
    return $strongest->abilities->speed >= $pokemon->abilities->speed;  
})  
->each->flee();
```



Utilisez les macros !

Trier les pokémons par aptitude

```
$strongest = $pokedex  
  ->sortByDesc(function (Pokemon $pokemon) {  
    return $pokemon->abilities->sum();  
  })  
  ->first();
```

Utilisez les macros !

Trier les pokémons par aptitude

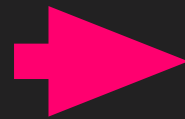
```
// app/Providers/CollectionServiceProvider@boot

Collection::macro('sortByAbility', function (string $ability = null) {
    return $this->sortByDesc(function (Pokemon $pokemon) use ($ability) {
        return $ability ? $pokemon->abilities->$ability : $pokemon->abilities->sum();
    });
});
```

Utilisez les macros !

Trier les pokémons par aptitude

```
$strongest = $pokedex  
->sortByDesc(function (Pokemon $pokemon) {  
    return $pokemon->abilities->sum();  
})  
->first();
```



```
$strongest = $pokedex->sortByAbility()->first();
```



Utilisez les macros !

Trier les pokémons par aptitude

```
// three strongest special attacks
$pokemons = $pokedex->sortByAbility('spAttack')->take(3);

// three strongest poison pokemons
$pokemons = $pokedex->filter->isType('poison')->sortByAbility()->take(3);

// weakest water pokemon
$pokemon = $pokedex->filter->isType('water')->sortByAbility()->last();
```


Conclusion

- ➔ Un peu de PF pour améliorer sa POO
- ➔ Langage multi-paradigmes - « Objet-fonctionnel »
- ➔ Package indépendant : <https://github.com/tightenco/collect>
- ➔ Pensez-y à votre prochain `foreach`

Références

- ➔ Documentation : <https://laravel.com/docs/master/collections>
- ➔ Package indépendant : <https://github.com/tightenco/collect>
- ➔ Livre : <https://adamwathan.me/refactoring-to-collections>
(-25% avec « laravelnews »)
- ➔ Dépôt Git : <https://github.com/osteel/laravel-collections-pres>
- ➔ BDD pokémons : <https://github.com/fanzeyi/pokemon.json>

Questions

