Collection 集合类

使用

CI中的使用

将

Collection.php

Collection/

拷贝到 application/libraries/ 下

在 application/config/autoload.php $autoload['libraries'] 中加入 'Collection'

方法

构建对象 make()

```
$collection = Collection::make([]);
```

或

```
$collection = new Collection([]);
```

添加元素（所添加的元素的键值元数据中必须不存在，存在则不会添加 add()

```
Collection::make([])->add('a', 1); // ["a":1]
```

将对象转为数组（获取对象的数据源）toArray()

```
Collection::make(['a'])->toArray(); // ["a"]
```

获取全部数据 all()

toArray() 的别名

获取数据源中指定字段的平均值 avg()

```
Collection::make([1,2,3,4,5])->avg(); // 3
```

二维数组中指定字段的平均值

```
Collection::make([
    ['value' => 1],
    ['value' => 2],
    ['value' => 3],
    ['value' => 4],
    ['value' => 5]
])->avg('value'); // 3
```

二维数组中不指定字段（将数组当做数字使用，会默认转为 0）

```
Collection::make([
    ['value' => 1],
    ['value' => 2],
    ['value' => 3],
    ['value' => 4],
    ['value' => 5]
])->avg(); // 0
```

第二个参数可传入回调函数以处理返回值

```
Collection::make([
    ['value' => 1],
    ['value' => 2],
    ['value' => 3],
    ['value' => 4]
])->avg('value', function($v) {
    return ceil($v);
}); // 3
```

将数组拼接到本数组后方，不考虑键 concat()

```
Collection::make([])->concat([1,2,3,5]); // [1,2,3,5]
//参数支持传入集合实例
```

集合是否包含指定值 contains()

```
Collection::make([1,23,4])->contains(23); // true
```

可以传入键值对

```
Collection::make(['a' => 1])->contains('a', 1); //true
Collection::make(['a' => 1])->contains('a', 2); //false
```

可以传入回调函数，判断是否包含符合回调函数的数据

```
Collection::make(['a' => 1])->contains(function($v) {
    return $v == 1;
}); //true
```

将二维数组合并为一维数组 collaspe()

```
Collection::make([[1,2,3],[4,5,6]])->collaspe(); // [1,2,3,4,5,6]
```

获取数组的键值对，可以和 list 共用 divide()

```
list($keys, $values) = Collection::make(['a' => 1, 'b' => 2])-
>divide();
// keys： ["a", "b"]
// values: [1,2]
```

将多维数组铺开到一维数组，使用 . 表示深度 dot()

```
Collection::make(['a' => ['b' => ['c' => 1]]])->dot(); // [
"a.b.c":1 ]
```

数据源中指定键自减 decrement()

```
Collection::make([1,2,3,4])->decrement(0); // [2,2,3,4]
```

数据源中指定键减少加指定值

```
Collection::make([1,2,3,4])->decrement(0, 2); // [3,2,3,4]
```

数据源中指定键不存在，创建并默认为0，自减1

```
Collection::make([1,2,3,4])->decrement(4); // [1,2,3,4,1]
```

遍历数组（函数返回 false 则跳出循环） each()

foreach 的别名

```
Collection::make([1,2,4,5])->each(function($v, $k) {
    echo $k , ' => ', $v, "\n";
});
// 0 => 1
// 1 => 2
// 2 => 4
// 3 => 5
```

迭代集合的元素，将每个嵌套元素值传递给给定的回调 eachSpread()

```
Collection::make([['John Doe', 35], ['Jane Doe', 33]])
->eachSpread(function ($name, $age) {
    //...
});
```

获取除指定键之外的元素 except()

```
Collection::make(['a' => 1, 'b' => 2])->except('a'); // ["b":1]
```

传入的参数可以为数组或者 Collection 实例

```
$except = ['a', 'b'];
$exceptCollection = Collection::make($except);

Collection::make(['a'=>1, 'b'=>2, 'c'=> 3])->except($except); //
["c":3]
Collection::make(['a'=>1, 'b'=>2, 'c'=> 3])-
>except($exceptCollection); // ["c":3]
```

获取数组的第一个元素 first()

```
Collection::make([1,23,45])->first(); // 1
```

第一个参数传入一个函数则返回第一个函数返回 true 的元素

```
Collection::make([1,2,3,5])->first(function($v) {
    return $v > 1;
}); // 2
```

第二个参数可以传入一个默认值，找不到符合要求的数据则返回默认值

```
Collection::make([1,2,3,5])->first(function($v) {
    return $v > 10;
}, 0); // 0
```

将数组铺开并重置键 flatten()

```
Collection::make(['a' => 1, 'b' => ['c' => 2]])->flatten(); //
[1,2]
```

移除指定键，可以使用 . 符号移除深层键 forget()

```
Collection::make(['a' => 1, 'b' => ['c' => 2]])->forget('b.c'); //
["a":1, "b":[]]
```

数据分页 forPage()

```
Collection::make([1,2,3,4,5,6,7,8,9,10])->forPage(0,2); //[1,2]
```

获取数据源中指定的键值 get()

```
Collection::make(['a' => 1])->get('a'); // 1
```

可以使用 . 表示分割获取深层数据

```
Collection::make(['a' => ['b' => ['c' => 2]]])->get('a.b.c'); // 2
```

如果数据源以数字为键

```
Collection::make(['a'])->get(0); // a
```

如果数据源为多维数组

```
Collection::make([
        'a' => ['a'],
        'b' => ['b'],
        'c' => ['c']
    ])->get(0); // ["a"]
```

方法的第二个参数为默认值，当要获取的键不存在时，将会返回 null，有默认值则会返回默认值

```
Collection::make(['a'])->get(1); // null
Collection::make(['a'])->get(1, 'default'); // default
```

如果不传参数则会返回整个数据源，相当于 `toArray()`

```
Collection::make(['a'])->get(); // ["a"]
```

将对象按照指定字段分组 groupBy()

```
//数据源
$array = [
    ['hour' => '10:00', 'flow' => 'aaa', 'value' => 10],
    ['hour' => '10:00', 'flow' => 'bbb', 'value' => 8],
    ['hour' => '10:30', 'flow' => 'aaa', 'value' => 6],
    ['hour' => '10:30', 'flow' => 'bbb', 'value' => 15],
    ['hour' => '11:00', 'flow' => 'aaa', 'value' => 13],
    ['hour' => '11:00', 'flow' => 'bbb', 'value' => 5],
    ['hour' => '11:30', 'flow' => 'aaa', 'value' => 9],
    ['hour' => '11:30', 'flow' => 'bbb', 'value' => 16],
];
```

以下所有代码都依赖于上述数据源

基本用法

```
Collection::make($array)->groupBy('flow');

//分组后的数据源，同组数据将组合为一个新的以数字为键的数组
//[
//    "aaa" => [
//        [ "hour" => "10:00", "flow" => "aaa", "value" => 10 ]
//        [ "hour" => "10:30", "flow" => "aaa", "value" => 6 ]
//        [ "hour" => "11:00", "flow" => "aaa", "value" => 13 ]
//        [ "hour" => "11:30", "flow" => "aaa", "value" => 9 ]
//    ]
//    "bbb" => [
//        [ "hour" => "10:00", "flow" => "bbb", "value" => 8 ]
//        [ "hour" => "10:30", "flow" => "bbb", "value" => 15 ]
//        [ "hour" => "11:00", "flow" => "bbb", "value" => 5 ]
//        [ "hour" => "11:30", "flow" => "bbb", "value" => 16 ]
//    ]
//]
```

回调处理，很多时候，我们不仅需要分组，还需要对分组之后的数据进行处理，你可以在回调函数中使用 Collection 的所有方法，不过请务必使用 return 返回

```
Collection::make($array)->groupBy('flow', function ($v) {
    return array_column($v, 'hour');
});
//[
//    [aaa] => [ "10:00", "10:30", "11:00", "11:30" ]
//    [bbb] => [ "10:00", "10:30", "11:00", "11:30" ]
//]
```

希望保留原有键，将第三个参数设为 true 即可

```php
Collection::make($array)->groupBy('flow', null, true);

//分组后的数据源，同组数据将组合为一个新的以原有数字为键的数组
//[
//    "aaa" => [
//        0 => [ "hour" => "10:00", "flow" => "aaa", "value" => 10 ]
//        1 => [ "hour" => "10:30", "flow" => "aaa", "value" => 6 ]
//        2 => [ "hour" => "11:00", "flow" => "aaa", "value" => 13 ]
//        3 => [ "hour" => "11:30", "flow" => "aaa", "value" => 9 ]
//    ]
//    "bbb" => [
//        4 => [ "hour" => "10:00", "flow" => "bbb", "value" => 8 ]
//        5 => [ "hour" => "10:30", "flow" => "bbb", "value" => 15 ]
//        6 => [ "hour" => "11:00", "flow" => "bbb", "value" => 5 ]
//        7 => [ "hour" => "11:30", "flow" => "bbb", "value" => 16 ]
//    ]
//]
```

按照多个字段进行分组，第一个参数传入数组

```
Collection::make($array)->groupBy(['flow', 'hour']);
//[
//    "aaa" => [
//        "10:00" => [
//            [ "hour" => "10:00", "flow" => "aaa", "value" => 10 ]
//        ]
//        "10:30" => [
//            [ "hour" => "10:30", "flow" => "aaa", "value" => 6 ]
//        ]
//        "11:00" => [
//            [ "hour" => "11:00", "flow" => "aaa", "value" => 13 ]
//        ]
//        "11:30" => [
//            [ "hour" => "11:30", "flow" => "aaa", "value" => 9 ]
//        ]
//    "bbb" => [
//        "10:00" => [
//            [ "hour" => "10:00", "flow" => "bbb", "value" => 8 ]
//        ]
//        "10:30" => [
//            [ "hour" => "10:30", "flow" => "bbb", "value" => 15 ]
//        ]
//        "11:00" => [
//            [ "hour" => "11:00", "flow" => "bbb", "value" => 5 ]
//        ]
//        "11:30" => [
//            [ "hour" => "11:30", "flow" => "bbb", "value" => 16 ]
//        ]
//    ]
//]
```

按照回调函数分组（可能需要按照指定字段的前两个字符分组）

```
Collection::make($array)->groupBy(function ($v) {
    return substr($v['hour'], 0, 2);
})
//[
//    10 => [
//        [ "hour" => "10:00", "flow" => "aaa", "value" => 10 ]
//        [ "hour" => "10:00", "flow" => "bbb", "value" => 8 ]
//        [ "hour" => "10:30", "flow" => "aaa", "value" => 6 ]
//        [ "hour" => "10:30", "flow" => "bbb", "value" => 15 ]
//    ]
//    11 => [
//        [ "hour" => "11:00", "flow" => "aaa", "value" => 13 ]
//        [ "hour" => "11:00", "flow" => "bbb", "value" => 5 ]
//        [ "hour" => "11:30", "flow" => "aaa", "value" => 9 ]
//        [ "hour" => "11:30", "flow" => "bbb", "value" => 16 ]
//    ]
//]
```

判断指定键是否存在，支持 . 语法 has()

```
Collection::make(['a' => ['b' => 1]])->has('a.b'); // true
```

数据源中指定键自增 increment()

```
Collection::make([1,2,3,4])->increment(0); // [2,2,3,4]
```

数据源中指定键增加指定值

```
Collection::make([1,2,3,4])->increment(0, 2); // [3,2,3,4]
```

数据源中指定键不存在，创建并默认为0，自增1

```
Collection::make([1,2,3,4])->increment(4); // [1,2,3,4,1]
```

数组拼接 implode()

```
Collection::make(['a', 'b'])->implode(','); // a,b
```

判断数组是否为空 isEmpty()

```
Collection::make([])->isEmpty(); // true
```

判断数组是否不为空 isNotEmpty()

```
Collection::make([])->isNotEmpty(); // false
```

将元素数组的某一项作为键，类似 groupBy()，但仅支持普通键以及函数参数，不支持数组参数 keyBy()

```
Collection::make($array)->groupBy('flow');

//分组后的数据源，同组数据将组合为一个新的以数字为键的数组
//[
//    "aaa" => [
//        [ "hour" => "10:00", "flow" => "aaa", "value" => 10 ]
//        [ "hour" => "10:30", "flow" => "aaa", "value" => 6 ]
//        [ "hour" => "11:00", "flow" => "aaa", "value" => 13 ]
//        [ "hour" => "11:30", "flow" => "aaa", "value" => 9 ]
//    ]
//    "bbb" => [
//        [ "hour" => "10:00", "flow" => "bbb", "value" => 8 ]
//        [ "hour" => "10:30", "flow" => "bbb", "value" => 15 ]
//        [ "hour" => "11:00", "flow" => "bbb", "value" => 5 ]
//        [ "hour" => "11:30", "flow" => "bbb", "value" => 16 ]
//    ]
//]
```

获取集合中最大值所对应的键名 keysOfMaxValue()

```
Collection::make(['a' => 1, 'b' => 2])->keysOfMaxValue(); // ['b']
```

获取数组的最后一个元素 last()

```
Collection::make([1,23,45])->last(); // 45
```

第一个参数传入一个函数则返回第一个函数返回 true 的元素

```
Collection::make([1,2,3,5])->last(function($v) {
    return $v < 3;
}); // 3
```

第二个参数可以传入一个默认值，找不到符合要求的数据则返回默认值

```
Collection::make([1,2,3,5])->last(function($v) {
    return $v < 0;
}, 10); // 10
```

仅获取指定字段 only()

```
Collection::make(['a' => 1, 'b' => 2])->only('a'); // ['a' => 1]
```

可以通过数组传入多个键名

```
Collection::make(['a' => 1, 'b' => 2])->only(['a', 'b']); // ['a'
=> 1, 'b' => 2]
```

获取二维数组指定键值，返回结果不包含键名 pluck()

```
Collection::make($array)->pluck('flow'); //
["aaa","bbb","aaa","bbb","aaa","bbb","aaa","bbb"]
```

在数组的最前方插入指定值 prepend()

```
Collection::make([1])->prepend(2); // [2,1]
```

第二个参数是传入值的键

```
Collection::make(['a' => 1])->prepend(2, 'b'); // ["b":2, "a":1]
```

移除指定键的元素并返回该元素 pull()

```
Collection::make(['a' => 1])->pull('a') // 1
```

设置数据源中指定的键值 set()

```
$collection = Collection::make(['a']); // ["a"]
$collection->set(0, 'b'); // ["b"]
```

可以使用 . 表示分割设置深层数据

```
$collection = Collection::make(['a' => ['b' => 2]]); // ["a":
["b":2]]
$collection->set('a.b', 3); // ["a":["b":3]]
```

如果目标键不存在，则会新建

```
$collection = Collection::make(['a' => ['b' => 2]]); // ["a":
["b":2]]
$collection->set('a.c', 3); // ["a":["b":2, "c":3]]
```

二维数组按照元素指定键升序排序 sortBy()

```
Collection::make($array)->sortBy('hour');
//[
//     [ "hour" => "10:00", "flow" => "aaa", "value" => 10 ]
//     [ "hour" => "10:00", "flow" => "bbb", "value" => 8  ]
//     [ "hour" => "10:30", "flow" => "aaa", "value" => 6  ]
//     [ "hour" => "10:30", "flow" => "bbb", "value" => 15 ]
//     [ "hour" => "11:00", "flow" => "aaa", "value" => 13 ]
//     [ "hour" => "11:00", "flow" => "bbb", "value" => 5  ]
//     [ "hour" => "11:30", "flow" => "aaa", "value" => 9  ]
//     [ "hour" => "11:30", "flow" => "bbb", "value" => 16 ]
//]
```

还可以按照传入的匿名函数进行排序

```
Collection::make($array)->sortBy(function($v) {
    return $v['hour'];
});
//[
//     [ "hour" => "10:00", "flow" => "aaa", "value" => 10 ]
//     [ "hour" => "10:00", "flow" => "bbb", "value" => 8  ]
//     [ "hour" => "10:30", "flow" => "aaa", "value" => 6  ]
//     [ "hour" => "10:30", "flow" => "bbb", "value" => 15 ]
//     [ "hour" => "11:00", "flow" => "aaa", "value" => 13 ]
//     [ "hour" => "11:00", "flow" => "bbb", "value" => 5  ]
//     [ "hour" => "11:30", "flow" => "aaa", "value" => 9  ]
//     [ "hour" => "11:30", "flow" => "bbb", "value" => 16 ]
//]
```

获取指定数量的数据 take()

```
Collection::make([1,2,34,5,6,7,89])->take(2); //[1,2]
```

将数组转为json toJson()

```
Collection::make(['a'=>1])->toJson(); // ["a":1]
```

当第一个参数为 false 时，将当前集合传入第二个参数（函数）中处理，为 true 则不处理 unless()

```
Collection::make([1,2])->unless(false, function($c) {
    echo $c->toJson(); // [1,2]
});
```

当第一个参数为 true 时，将当前集合传入第二个参数（函数）中处理，为 false 则不处理 when()

```
Collection::make([1,2])->when(true, function($c) {
    echo $c->toJson(); // [1,2]
});
```

过滤出符合要求的数据 where()

```
//数据源
$array = [
    ['hour' => '10:00', 'flow' => 'aaa', 'value' => 10],
    ['hour' => '10:00', 'flow' => 'bbb', 'value' => 8],
    ['hour' => '10:30', 'flow' => 'aaa', 'value' => 6],
    ['hour' => '10:30', 'flow' => 'bbb', 'value' => 15],
    ['hour' => '11:00', 'flow' => 'aaa', 'value' => 13],
    ['hour' => '11:00', 'flow' => 'bbb', 'value' => 5],
    ['hour' => '11:30', 'flow' => 'aaa', 'value' => 9],
    ['hour' => '11:30', 'flow' => 'bbb', 'value' => 16],
];

Collection::make($array)->where('flow', 'aaa');
//[
//     ['hour' => '10:00', 'flow' => 'aaa', 'value' => 10],
//     ['hour' => '10:30', 'flow' => 'aaa', 'value' => 6],
//     ['hour' => '11:00', 'flow' => 'aaa', 'value' => 13],
//     ['hour' => '11:30', 'flow' => 'aaa', 'value' => 9],
//]
```

可以传入比较符 = > < >= <= == !=

```
Collection::make($array)->where('flow', '=', 'aaa');
//[
//    ['hour' => '10:00', 'flow' => 'aaa', 'value' => 10],
//    ['hour' => '10:30', 'flow' => 'aaa', 'value' => 6],
//    ['hour' => '11:00', 'flow' => 'aaa', 'value' => 13],
//    ['hour' => '11:30', 'flow' => 'aaa', 'value' => 9],
//]
```

可以传入数组，表示必须符合多个条件

```
Collection::make($array)->where([['flow', '=', 'aaa'], ['hour',
'10:00']]);
//[
//    ['hour' => '10:00', 'flow' => 'aaa', 'value' => 10]
//]
```

过滤出指定键值在给定值中的记录 whereIn()

```
Collection::make($array)->whereIn('flow', ['aaa']);
//[
//    ['hour' => '10:00', 'flow' => 'aaa', 'value' => 10],
//    ['hour' => '10:30', 'flow' => 'aaa', 'value' => 6],
//    ['hour' => '11:00', 'flow' => 'aaa', 'value' => 13],
//    ['hour' => '11:30', 'flow' => 'aaa', 'value' => 9],
//]
```

封装了全部的 PHP 原生数组函数

原生	封装	简化
原生的 PHP 方法	通过 _ 分割，再以小驼峰命名	除去开头的 array 再以小驼峰命名
array_filter()	$collection->arrayFilter()	$collection->filter()