55．验证规则和错误信息

学习要点：

1.验证规则

2.错误信息

本节课我们来学习一下数据验证的知识，这节了解验证规则和错误信息。

一．验证规则

1． 在上一节验证器定义的时候，我们采用的字符串模式，也支持数组模式；

```
protected $rule = [
    'name'      =>      [
        'require',
        'max'       =>      10,
        'checkName' =>      '李炎恢'
      ],
      'price'     =>      [
        'number',
        'between'   =>      '1,100'
      ],
      'email'     =>      'email'
   ];
```

2． 数组模式在验证规则很多很乱的情况下，更容易管理，可读性更高；

3． 如果你想使用独立验证，就是手动调用验证类，而不是调用 User.php 验证类；

4． 这种调用方式，一般来说，就是独立、唯一，并不共享的调用方式；

```
$validate = Validate::rule([
    'name'      =>      'require|max:20',
    'price'     =>      'number|between:1,100',
    'email'     =>      'email'
]);

$result = $validate->check([
    'name'  =>  '李炎恢',
    'price' =>  90,
    'email' =>  'xiaoxin163.com'
]);

if (!$result) {
    dump($validate->getError());
}
```

5. 独立验证默认也是返回一条错误信息，如果要批量返回所有错误使用 batch()；

```
$result = $validate->batch(true)->check
```

6. 独立验支持对象化的定义方式，但不支持在属性方式的定义；

```
$validate = Validate::rule([
    'name'      =>      ValidateRule::isRequire()->max(20),
    'price'     =>      ValidateRule::isNumber()->between([1, 100]),
    'email'     =>      ValidateRule::isEmail()
]);
```

8. 独立验支持闭包的自定义方式，但这种方式会不支持字段的多规则；

```
$validate = Validate::rule([
    'name' => function ($value) {
        return $value != '' ? true : '姓名不得为空';
    },
    'price'=> function ($value) {
        return $value > 0 ? true : '价格不得小于零';
    }
]);
```

二．错误信息

1. 独立验证的自定义错误提示，可以在方法的第二参数，参数一是规则；

```
ValidateRule::isEmail(null, '邮箱格式不正确！');
ValidateRule::isNumber()->between([1, 100], '价格范围 1-100 之间')
```

2. 也可以独立使用 message()方法，来设置相关错误信息；

```
$validate->message([
    'name.require'      =>      '姓名不得为空',
    'name.max'          =>      '姓名不可以超过 20 个子'
]);

'name.require'      =>      ['code'=>1001, 'msg'=>'姓名不得为空'],
```