Validation

simple

```
$this->validate($request, [
    'title' => 'required|unique:posts|max:255',
    'author.name' => 'required',
    'author.description' => 'required',
]);
```

bail stop running validation rules on an attribute after the first validation failure

内嵌的参数 使用 点 google 了一下就是form 里面 name = author['name']

```
$this->validate($request, [
    'title' => 'required|unique:posts|max:255',
    'author.name' => 'required',
    'author.description' => 'required',
]);
```

you will often need to mark your "optional" request fields as nullable if you do not want the validator to consider nullvalues as invalid
默认的Middleware (查看App\Http\Kernel.php)会将空变成null 如果没有 nullable 的话有可能会出错

```
#显示error
<h1>Create Post</h1>
@if (count($errors) > 0)
    <div class="alert alert-danger">
        <ul>
            @foreach ($errors->all() as $error)
                <li>{{ $error }}</li>
            @endforeach
        </ul>
    </div>
@endif
```

```
#自定义错误
overwrite Controller 里面的 formatValidationErrors 方法
protected function formatValidationErrors(Validator $validator)
{
    return $validator->errors()->all();
}
```

AJAX Requests & Validation
When using the validate method during an AJAX request, Laravel will not generate a redirect response 将会返回一个json with 422 HTTP status code

创建一个特定的Form Request Validation
`php artisan make:request StoreBlogPost`

```
public function rules()
{
    return [
        'title' => 'required|unique:posts|max:255',
        'body' => 'required',
    ];
}
```

add After Hooks To Form Requests

```
withValidator 使用这个方法 添加另外的验证错误信息
public function withValidator($validator)
{
    $validator->after(function ($validator) {
        if ($this->somethingElseIsInvalid()) {
            $validator->errors()->add('field', 'Something is wrong with this field!');
        }
    });
}
```

authorize函数 用来验证是否登录的用户能够请求该request

```
public function authorize()
{
    $comment = Comment::find($this->route('comment'));
    return $comment && $this->user()->can('update', $comment);
}
```

自定义The Error Format 在自定义的request 里面 覆盖formatErrors方法

```
protected function formatErrors(Validator $validator)
{
    return $validator->errors()->all();
}
```

自定义错误信息

```
public function messages()
{
    return [
        'title.required' => 'A title is required',
        'body.required'  => 'A message is required',
    ];
}
```

Manually Creating Validators

在Controller中创建自定义的Validator
#The first argument passed to the `make` method is the data under validation. The second argument is the validation rules that should be applied to the data.

```
use Validator;
public function store(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'title' => 'required|unique:posts|max:255',
            'body' => 'required',
        ]);

        if ($validator->fails()) {
            return redirect('post/create')
                        ->withErrors($validator)
                        ->withInput($request->except('password'));
        }

        // Store the blog post...
    }
```

```
假如还要使用原来validate的redirect 功能 可以这样
Validator::make($request->all(), [
    'title' => 'required|unique:posts|max:255',
    'body' => 'required',
])->validate();
```

```
假如有多个from 在一个页面上面的时候 需要 named error bags
return redirect('register')
            ->withErrors($validator, 'login');
```

在blade文件中{{ $errors->login->first('email') }}

```
# after validation hook
$validator->after(function ($validator) {
    if ($this->somethingElseIsInvalid()) {
        $validator->errors()->add('field', 'Something is wrong with this field!');
    }
});
```

获取错误信息

```
#获得第一个参数
$errors = $validator->errors();
echo $errors->first('email');
```

```
获取一个字段的所有错误
foreach ($errors->get('email') as $message) {
}
```

```
如果是数组形式的name
foreach ($errors->get('attachments.*') as $message) {
}
```

```
获取所有error
$errors->all()
```

```
判断某个字段的错误信息是否存在
$errors->has('email')
```

```
自定义错误信息
$messages = [
    'required' => 'The :attribute field is required.',
];

$validator = Validator::make($input, $rules, $messages);
```

```
$messages = [
    'same'    => 'The :attribute and :other must match.',
    'size'    => 'The :attribute must be exactly :size.',
    'between' => 'The :attribute must be between :min - :max.',
    'in'      => 'The :attribute must be one of the following types: :values',
];
如果想指定某种特殊的错误信息 比如 email 必须 required 错误信息
'email.required' => "email is  required 哈哈哈"
```

```
email is required 哈哈哈
The password is really really really important.
The password confirm is really really really important.
```

如果有多语言平台 可以将提示语句写在
To do so, add your messages to custom array in the resources/lang/xx/validation.php language file

```
'custom' => [
    'email' => [
        'required' => 'We need to know your e-mail address!',
    ],
],
```

自定义 错误message 中 :attribute 在 resources/lang/xx/validation.php 文件中 有一个 'attributes' => [
    'email' => 'email address',
],

```
The english name is really really really important.
The email must be a valid email address.
The password confirm is really really really important.
```

所有规则 https://laravel.com/docs/5.4/validation#available-validation-rules