**AngularJs之表单及表单验证**

## ng中的表单与Controller

　　看这个小标题也行你会差异，[表单](http://cpro.baidu.com/cpro/ui/uijs.php?c=news&cf=1001&ch=0&di=128&fv=17&jk=cc208e80747141eb&k=%B1%ED%B5%A5&k0=%B1%ED%B5%A5&kdi0=0&luki=2&n=10&p=baidu&q=xiaoxiaobai_cpr&rb=0&rs=1&seller_id=1&sid=eb417174808e20cc&ssp2=1&stid=0&t=tpclicked3_hc&tu=u1972324&u=http%3A%2F%2Fwww%2E52ij%2Ecom%2Fjishu%2F5478%2Ehtml&urlid=0)验证，怎么跟controller扯上关系了。ng中的form已经不同于我们平时用的form标签，做了增强。**form是FormController的一个实例**。如何理解这句话呢？想想我们使用ng-controller指令的情景：

<div ng-controller="testC">

<input type="test" ng-[model](http://cpro.baidu.com/cpro/ui/uijs.php?c=news&cf=1001&ch=0&di=128&fv=17&jk=cc208e80747141eb&k=model&k0=model&kdi0=0&luki=10&n=10&p=baidu&q=xiaoxiaobai_cpr&rb=0&rs=1&seller_id=1&sid=eb417174808e20cc&ssp2=1&stid=0&t=tpclicked3_hc&tu=u1972324&u=http%3A%2F%2Fwww%2E52ij%2Ecom%2Fjishu%2F5478%2Ehtml&urlid=0)="a" />

</div>

<scritp> function testC($scope){ //............. } </script>

　　应用了ng-controller的div就是testC的一个实例，我们可以在模板中使用定义在$scopt上的任何属性和方法，而 testC的定义也是由我们自己实现的。当我们使用<form>的时候也是这样的道理，FormController由ng为我们定义好了， 有一系列属性和方法提供给我们完成验证工作，form实例通过name属性来进行[标识](http://cpro.baidu.com/cpro/ui/uijs.php?c=news&cf=1001&ch=0&di=128&fv=17&jk=cc208e80747141eb&k=%B1%EA%CA%B6&k0=%B1%EA%CA%B6&kdi0=0&luki=6&n=10&p=baidu&q=xiaoxiaobai_cpr&rb=0&rs=1&seller_id=1&sid=eb417174808e20cc&ssp2=1&stid=0&t=tpclicked3_hc&tu=u1972324&u=http%3A%2F%2Fwww%2E52ij%2Ecom%2Fjishu%2F5478%2Ehtml&urlid=0)，我们可以通过此标识来访问form实例的属性和方法，如：

<form name="myform"> {{myform.$valid}} </form>

　　form提供的属性都是用来表示表单的验证状态的，包括：$pristine(表单没有填写记录)、$dirty(表单有填写记 录)、$valid(通过验证)、$invalid(未通过验证)、$error(验证错误信息)。除$error外，前四个的值为true或false 表示相应的状态。$error的值为一个js对象，包含了以下验证内容的状态：

* email
* max
* maxlength
* min
* minlength
* number
* pattern
* required
* url

　　这些内容我们会在稍后的例子中看到。FormController还提供了一些方法，我们一般不手工调用它们，都是系统自己调用。可参考官方文档：<http://docs.angularjs.org/api/ng.directive:form.FormController>

　　表单元素，如input、checkbox、radio等也不是普通的表单元素了，它们通通是NgModelController的实例。与 form一样，也是通过name属性来标识。FormController拥有的那五个属性，NgModelController也同样拥有，除此之外， 还有许多额外的属性和方法，我们稍后也在示例中展示，可参考官方文档：<http://docs.angularjs.org/api/ng.directive:ngModel.NgModelController>

　　还有一个特性需要了解，一个[表单](http://cpro.baidu.com/cpro/ui/uijs.php?c=news&cf=1001&ch=0&di=128&fv=17&jk=cc208e80747141eb&k=%B1%ED%B5%A5&k0=%B1%ED%B5%A5&kdi0=0&luki=2&n=10&p=baidu&q=xiaoxiaobai_cpr&rb=0&rs=1&seller_id=1&sid=eb417174808e20cc&ssp2=1&stid=0&t=tpclicked3_hc&tu=u1972324&u=http%3A%2F%2Fwww%2E52ij%2Ecom%2Fjishu%2F5478%2Ehtml&urlid=0)中的表单元素，会作为这个form的属性自动加在上面，通过name[标识](http://cpro.baidu.com/cpro/ui/uijs.php?c=news&cf=1001&ch=0&di=128&fv=17&jk=cc208e80747141eb&k=%B1%EA%CA%B6&k0=%B1%EA%CA%B6&kdi0=0&luki=6&n=10&p=baidu&q=xiaoxiaobai_cpr&rb=0&rs=1&seller_id=1&sid=eb417174808e20cc&ssp2=1&stid=0&t=tpclicked3_hc&tu=u1972324&u=http%3A%2F%2Fwww%2E52ij%2Ecom%2Fjishu%2F5478%2Ehtml&urlid=0)就可以访问到，如：

<form name="myform">

<input type="text" name="myname" /> {{myform.myname.$valid}} </form>

## ng内置的验证规则

　　ng框架提供了非常方便的验证机制，你只需要在标签上加点指令，像使用HTML5提供的验证那样，然后在[css](http://cpro.baidu.com/cpro/ui/uijs.php?c=news&cf=1001&ch=0&di=128&fv=17&jk=cc208e80747141eb&k=css&k0=css&kdi0=0&luki=7&n=10&p=baidu&q=xiaoxiaobai_cpr&rb=0&rs=1&seller_id=1&sid=eb417174808e20cc&ssp2=1&stid=0&t=tpclicked3_hc&tu=u1972324&u=http%3A%2F%2Fwww%2E52ij%2Ecom%2Fjishu%2F5478%2Ehtml&urlid=0)中根据规则定义好正确/错误的样式就OK了，例如我们要让一个文本框为必填项，使用required：

<form name="myform novalidate> <input type="text" ng-model="a" required />

</form>

　　有几点需要注意：

1. 在<form>上加了一个novalidate，用来禁止掉浏览器默认的验证行为，因为ng已经对HTML5的几种表单新特性做了兼容处理。
2. 表单元素必须有ng-[model](http://cpro.baidu.com/cpro/ui/uijs.php?c=news&cf=1001&ch=0&di=128&fv=17&jk=cc208e80747141eb&k=model&k0=model&kdi0=0&luki=10&n=10&p=baidu&q=xiaoxiaobai_cpr&rb=0&rs=1&seller_id=1&sid=eb417174808e20cc&ssp2=1&stid=0&t=tpclicked3_hc&tu=u1972324&u=http%3A%2F%2Fwww%2E52ij%2Ecom%2Fjishu%2F5478%2Ehtml&urlid=0)，否则无法触发验证
3. 在[css](http://cpro.baidu.com/cpro/ui/uijs.php?c=news&cf=1001&ch=0&di=128&fv=17&jk=cc208e80747141eb&k=css&k0=css&kdi0=0&luki=7&n=10&p=baidu&q=xiaoxiaobai_cpr&rb=0&rs=1&seller_id=1&sid=eb417174808e20cc&ssp2=1&stid=0&t=tpclicked3_hc&tu=u1972324&u=http%3A%2F%2Fwww%2E52ij%2Ecom%2Fjishu%2F5478%2Ehtml&urlid=0)中分别定义.ng-pristine、.ng-valid、.ng-invalid、.ng-dirty这四种样式，ng会根据相应的状态自动加上样式。

　　这部分还是相当简单的，下面我们写例子来测一下这几种验证机制，HTML代码如下：

<div ng-app="MyApp">

<div ng-controller="testC">

<form name="myform" novalidate>

required: <input type="text" name="test1" ng-model="test1" required><br />

ng-minlength(3): <input type="text" name="test2" ng-[model](http://cpro.baidu.com/cpro/ui/uijs.php?c=news&cf=1001&ch=0&di=128&fv=17&jk=cc208e80747141eb&k=model&k0=model&kdi0=0&luki=10&n=10&p=baidu&q=xiaoxiaobai_cpr&rb=0&rs=1&seller_id=1&sid=eb417174808e20cc&ssp2=1&stid=0&t=tpclicked3_hc&tu=u1972324&u=http%3A%2F%2Fwww%2E52ij%2Ecom%2Fjishu%2F5478%2Ehtml&urlid=0)="test2" ng-minlength="3"><br />

ng-maxlength(10):<input type="text" name="test3" ng-model="test3" ng-maxlength="10"><br />

ng-pattern(/[a-f]/): <input type="text" name="test4" ng-model="test4" ng-pattern="/[a-f]/"><br />

type="number"(2-8): <input type="number" name="test5" max="8" min="2" ng-model="test5"><br />

type="url": <input type="url" name="test6" ng-model="test6"><br />

type="email": <input type="email" name="test7" ng-[model](http://cpro.baidu.com/cpro/ui/uijs.php?c=news&cf=1001&ch=0&di=128&fv=17&jk=cc208e80747141eb&k=model&k0=model&kdi0=0&luki=10&n=10&p=baidu&q=xiaoxiaobai_cpr&rb=0&rs=1&seller_id=1&sid=eb417174808e20cc&ssp2=1&stid=0&t=tpclicked3_hc&tu=u1972324&u=http%3A%2F%2Fwww%2E52ij%2Ecom%2Fjishu%2F5478%2Ehtml&urlid=0)="test7"><br />

</form>

<div>

<h2>表单验证结果：</h2>

myform.$invalid : {{myform.$invalid}}<br />

myform.$valid : {{myform.$valid}}<br />

myform.$pristine : {{myform.$pristine}}<br />

myform.$dirty : {{myform.$dirty}}<br />

myform.$error : {{myform.$error}}<br />

<h2>[表单](http://cpro.baidu.com/cpro/ui/uijs.php?c=news&cf=1001&ch=0&di=128&fv=17&jk=cc208e80747141eb&k=%B1%ED%B5%A5&k0=%B1%ED%B5%A5&kdi0=0&luki=2&n=10&p=baidu&q=xiaoxiaobai_cpr&rb=0&rs=1&seller_id=1&sid=eb417174808e20cc&ssp2=1&stid=0&t=tpclicked3_hc&tu=u1972324&u=http%3A%2F%2Fwww%2E52ij%2Ecom%2Fjishu%2F5478%2Ehtml&urlid=0)项验证结果</h2>

required:<br />

myform.test1.$invalid : {{myform.test1.$invalid}}<br />

myform.test1.$valid : {{myform.test1.$valid}}<br />

myform.test1.$pristine : {{myform.test1.$pristine}}<br />

myform.test1.$dirty : {{myform.test1.$dirty}}<br />

myform.test1.$error : {{myform.test1.$error}}<br />

myform.test2.$error : {{myform.test2.$error}}<br />

</div>

</div>

</div>

CSS代码，为不同的状态设置不同的背景色：

input.ng-pristine { background-color: white;

} input.ng-dirty { background-color: lightyellow;

} input.ng-valid { background-color: lightgreen;

} input.ng-invalid { background-color: pink;

}

　　js代码，进行controller的初始化：

var app = angular.module('MyApp',[]);

app.controller('testC',function($scope){

$scope.test1='';

$scope.test2='';

$scope.test3='';

$scope.test4='';

$scope.test5='';

$scope.test6='';

$scope.test7='';

});

　　结果如下：

 　　该示例编写在runjs上，点击查看<http://runjs.cn/code/gspvlfrw>

　　在上面的代码中，你也看到了我从FormConroller实例myform访问到的属性，还有从NgModelController访问到的 属性。这些属性是非常有用的，比如你可以给表单的按钮加上：ng-disabled="myform.$invalid"，这样在表单未通过验证的时候， 提交按钮始终是不可点的。另外也可以根据表单元素的这些属性，来控制具体的错误提示信息，比如邮箱输错了，让"请输入正确的邮箱“这行字显示出来，如果你 顺着我的思路，应该立马能想象到。

## 自定义验证规则

　　除了内置的这些验证规则，你还可以自己定义。方法就是写一个指令，加在[表单](http://cpro.baidu.com/cpro/ui/uijs.php?c=news&cf=1001&ch=0&di=128&fv=17&jk=cc208e80747141eb&k=%B1%ED%B5%A5&k0=%B1%ED%B5%A5&kdi0=0&luki=2&n=10&p=baidu&q=xiaoxiaobai_cpr&rb=0&rs=1&seller_id=1&sid=eb417174808e20cc&ssp2=1&stid=0&t=tpclicked3_hc&tu=u1972324&u=http%3A%2F%2Fwww%2E52ij%2Ecom%2Fjishu%2F5478%2Ehtml&urlid=0)元素上。听起来好简单的样子，但是这指令与一般的指令可不同，我们需要按一定的规则来写，这样才可以融入ng的验证机制，让你自定义的跟内置的一样可以便捷的工作和管理。这个时候，NgModelController提供的方法就派上用场了。我们从一个例子来开始吧。

　　我想让我的输入框只允许输入偶数，我们来定义一个名为even-num的指令，在页面上使用的时候像这样：

<input type="number" ng-[model](http://cpro.baidu.com/cpro/ui/uijs.php?c=news&cf=1001&ch=0&di=128&fv=17&jk=cc208e80747141eb&k=model&k0=model&kdi0=0&luki=10&n=10&p=baidu&q=xiaoxiaobai_cpr&rb=0&rs=1&seller_id=1&sid=eb417174808e20cc&ssp2=1&stid=0&t=tpclicked3_hc&tu=u1972324&u=http%3A%2F%2Fwww%2E52ij%2Ecom%2Fjishu%2F5478%2Ehtml&urlid=0)="test1" even-num />

　　完整的js代码如下：

var app = angular.module('MyApp',[]);

app.controller('testC',function($scope){

$scope.test1 = '';

});

app.directive('evenNum',function(){

return {

require: 'ngModel',

link: function(scope, elm, attrs, ctrl)

{

ctrl.$parsers.push(function(viewValue) {

if (viewValue % 2 == 0) {

ctrl.$setValidity('evenNum', true);

return viewValue;

} else {

ctrl.$setValidity('evenNum', false);

return viewValue;

}

});

}

};

});

　　运行结果：

　　上面的例子编写在runjs上，点击查看<http://runjs.cn/code/gdq8m0gb>

　　现在来解释一下上面的代码。自定义指令的方式如果你不熟悉，可以先看一下我之前写的自定义指令部分。因为我们的指令要依赖 NgModelController，所以写上了require:'ngModel'，注意书写方式。 另外在link函数中，通过ctrl引用到了我们 注入的NgModelController，然后向它的$parsers属性中push了一个[函数](http://cpro.baidu.com/cpro/ui/uijs.php?c=news&cf=1001&ch=0&di=128&fv=17&jk=cc208e80747141eb&k=%BA%AF%CA%FD&k0=%BA%AF%CA%FD&kdi0=0&luki=9&n=10&p=baidu&q=xiaoxiaobai_cpr&rb=0&rs=1&seller_id=1&sid=eb417174808e20cc&ssp2=1&stid=0&t=tpclicked3_hc&tu=u1972324&u=http%3A%2F%2Fwww%2E52ij%2Ecom%2Fjishu%2F5478%2Ehtml&urlid=0)进去。这个$parsers是什么东西呢？很明显它是一个数组，因为我们可以push东西进去。在解释之前，我们先清楚两个概念：我们把模板中的数据，像{{aa}}这样的，叫做viewValue，故名思义，视图中的数据。我们把[模型](http://cpro.baidu.com/cpro/ui/uijs.php?c=news&cf=1001&ch=0&di=128&fv=17&jk=cc208e80747141eb&k=%C4%A3%D0%CD&k0=%C4%A3%D0%CD&kdi0=0&luki=4&n=10&p=baidu&q=xiaoxiaobai_cpr&rb=0&rs=1&seller_id=1&sid=eb417174808e20cc&ssp2=1&stid=0&t=tpclicked3_hc&tu=u1972324&u=http%3A%2F%2Fwww%2E52ij%2Ecom%2Fjishu%2F5478%2Ehtml&urlid=0)/controller中的数据，叫做[model](http://cpro.baidu.com/cpro/ui/uijs.php?c=news&cf=1001&ch=0&di=128&fv=17&jk=cc208e80747141eb&k=model&k0=model&kdi0=0&luki=10&n=10&p=baidu&q=xiaoxiaobai_cpr&rb=0&rs=1&seller_id=1&sid=eb417174808e20cc&ssp2=1&stid=0&t=tpclicked3_hc&tu=u1972324&u=http%3A%2F%2Fwww%2E52ij%2Ecom%2Fjishu%2F5478%2Ehtml&urlid=0)Value。 ng所说的双向绑定，就是把这两者进行绑定。这个$parsers保存了从viewValue向modelValue绑定过程中的处理函数，它们将来会依 次执行。因为我们的验证是从用户输入开始，即view发生了变化，所以我们的验证逻辑就加在这里。在验证结果中，我们调用 ctrl.$setValidity方法，将结果保存，这样框架就能完成接下来的一系列工作。

　　与$parsers相对的，还有一个属性叫$formatters，它保存的是从modelValue向viewValue绑定过程中的处理函 数。那我们定义的这个验证函数，要不要也push进$formatters里去呢？这取决于你的需要。如果你对两者的区别还不太清楚，看了下面这个例子就 明白了：

　　上面的例子编写在runjs上，点击查看<http://runjs.cn/code/9vde2r0w>

　　两个input的ng-model指向的是同一个，所以数据会同步变化，但是$formatters里没有push进去验证[函数](http://cpro.baidu.com/cpro/ui/uijs.php?c=news&cf=1001&ch=0&di=128&fv=17&jk=cc208e80747141eb&k=%BA%AF%CA%FD&k0=%BA%AF%CA%FD&kdi0=0&luki=9&n=10&p=baidu&q=xiaoxiaobai_cpr&rb=0&rs=1&seller_id=1&sid=eb417174808e20cc&ssp2=1&stid=0&t=tpclicked3_hc&tu=u1972324&u=http%3A%2F%2Fwww%2E52ij%2Ecom%2Fjishu%2F5478%2Ehtml&urlid=0)，所以在从modelValue向ViewValue绑定的过程中，副本并没有进行验证。如果把验证函数push进$formatters，那么副本也会跟着验证了。

## 自定义表单元素

　　我们都知道，在[表单](http://cpro.baidu.com/cpro/ui/uijs.php?c=news&cf=1001&ch=0&di=128&fv=17&jk=cc208e80747141eb&k=%B1%ED%B5%A5&k0=%B1%ED%B5%A5&kdi0=0&luki=2&n=10&p=baidu&q=xiaoxiaobai_cpr&rb=0&rs=1&seller_id=1&sid=eb417174808e20cc&ssp2=1&stid=0&t=tpclicked3_hc&tu=u1972324&u=http%3A%2F%2Fwww%2E52ij%2Ecom%2Fjishu%2F5478%2Ehtml&urlid=0)元素上使用ng-[model](http://cpro.baidu.com/cpro/ui/uijs.php?c=news&cf=1001&ch=0&di=128&fv=17&jk=cc208e80747141eb&k=model&k0=model&kdi0=0&luki=10&n=10&p=baidu&q=xiaoxiaobai_cpr&rb=0&rs=1&seller_id=1&sid=eb417174808e20cc&ssp2=1&stid=0&t=tpclicked3_hc&tu=u1972324&u=http%3A%2F%2Fwww%2E52ij%2Ecom%2Fjishu%2F5478%2Ehtml&urlid=0)可 以进行双向绑定。但是双向绑定只能用于input、checkbox这些标准表单控件上，你给一个div加ng-model是不能双向绑定的，因为系统不 知道该如何绑定。所以话说过来，要想给非标准表单控件双向绑定，代码还得自己来写，说白了就是自定义一个指令。其实这部分内容放在自定义指令中也是合适 的，但是官网在这里提到了，我也来介绍一下。

　　我们直接从例子开始，大家一定见过自适应的文本区域，就是随着输入内容的增加，会自动变高的textarea。用<textarea>标签做的话，需要加js代码才可以实现。更好的是纯[css](http://cpro.baidu.com/cpro/ui/uijs.php?c=news&cf=1001&ch=0&di=128&fv=17&jk=cc208e80747141eb&k=css&k0=css&kdi0=0&luki=7&n=10&p=baidu&q=xiaoxiaobai_cpr&rb=0&rs=1&seller_id=1&sid=eb417174808e20cc&ssp2=1&stid=0&t=tpclicked3_hc&tu=u1972324&u=http%3A%2F%2Fwww%2E52ij%2Ecom%2Fjishu%2F5478%2Ehtml&urlid=0)的方案，使用HTML5的新属性contenteditable，用一个div来模拟文本区域，div的高度默认就是自适应的，正好可以满足需求。基本的HTML代码和css代码如下：

<style>

.smarttextarea{

width: 400px;

min-height: 100px;

max-height: 400px;

border: 1px solid;

overflow: [auto](http://cpro.baidu.com/cpro/ui/uijs.php?c=news&cf=1001&ch=0&di=128&fv=17&jk=cc208e80747141eb&k=auto&k0=auto&kdi0=0&luki=5&n=10&p=baidu&q=xiaoxiaobai_cpr&rb=0&rs=1&seller_id=1&sid=eb417174808e20cc&ssp2=1&stid=0&t=tpclicked3_hc&tu=u1972324&u=http%3A%2F%2Fwww%2E52ij%2Ecom%2Fjishu%2F5478%2Ehtml&urlid=0);

padding:5px 10px 20px;

}

</[style](http://cpro.baidu.com/cpro/ui/uijs.php?c=news&cf=1001&ch=0&di=128&fv=17&jk=cc208e80747141eb&k=style&k0=style&kdi0=0&luki=3&n=10&p=baidu&q=xiaoxiaobai_cpr&rb=0&rs=1&seller_id=1&sid=eb417174808e20cc&ssp2=1&stid=0&t=tpclicked3_hc&tu=u1972324&u=http%3A%2F%2Fwww%2E52ij%2Ecom%2Fjishu%2F5478%2Ehtml&urlid=0)>

<div contenteditable=”true" class="smarttextarea"></div>

　　我们现在要做的就是，让这个模拟出来的文本区域跟真正的textarea那样，可以进行数据的双向绑定，这样就可以进行验证了。我定义了一个名为smarttextarea的指令，使用起来像这样：

<smarttextarea contenteditable="true" class="smarttextarea" ng-model="test3" required></smarttextarea>

指令的定义如下：

app.directive('smarttextarea',function(){

var link = function(scope, elm, attrs, ctrl) {

//view=>model数据绑定

elm.bind('keyup', function() {

scope.$apply(function() {

ctrl.$setViewValue(elm.html());

});

});

//[model](http://cpro.baidu.com/cpro/ui/uijs.php?c=news&cf=1001&ch=0&di=128&fv=17&jk=cc208e80747141eb&k=model&k0=model&kdi0=0&luki=10&n=10&p=baidu&q=xiaoxiaobai_cpr&rb=0&rs=1&seller_id=1&sid=eb417174808e20cc&ssp2=1&stid=0&t=tpclicked3_hc&tu=u1972324&u=http%3A%2F%2Fwww%2E52ij%2Ecom%2Fjishu%2F5478%2Ehtml&urlid=0)=>view数据绑定

ctrl.$render = function() {

elm.html(ctrl.$viewValue);

};

ctrl.$setViewValue(elm.html());

};

return {

template : '<div></div>',

replace : true,

require: 'ngModel',

restrict: 'E',

link : link

};

});

　　看一下效果：

　　上面的例子编写在runjs上，点击查看<http://runjs.cn/code/lhysp5vh>

　　我给模拟出来的textarea加了required验证，可以发现生效了。其实关键代码就是进行了数据的双向绑定处理，包括两步：

1. 从view向model绑定，监听keyup事件，然后调用ctrl.$setViewValue方法把viewValue保存下来
2. 从model想view绑定，调用ctrl.$render方法，将viewValue渲染到页面上

　　经过这两步，我们就自定义了一个跟标准表单控件一样的元素，可以进行数据的双向绑定，表单验证通通没有问题。本文来源 我爱IT技术网 http://www.52ij.com/jishu/5478.html 转载请保留链接。

# AngularJS 的表单验证

### 英文原文：[AngularJS Form Validation](http://scotch.io/tutorials/javascript/angularjs-form-validation)

**标签：** [AngularJS](http://www.oschina.net/translate/tag/angularjs)

14人收藏此文章, [我要收藏](javascript:add_to_favor(2213,7)) [oschina](http://my.oschina.net/osadmin) 推荐于 11个月前 (共 10 段, 翻译完成于 06-23) ([5评](http://www.oschina.net/translate/angularjs-form-validation#comments))

参与翻译(3人)：

[leoxu](http://my.oschina.net/xuleo), [petert](http://my.oschina.net/u/1422355), [AndyLam](http://my.oschina.net/andylam2010)

[仅中文](http://www.oschina.net/translate/angularjs-form-validation) | [中英文对照](http://www.oschina.net/translate/angularjs-form-validation?cmp) | [仅英文](http://www.oschina.net/translate/angularjs-form-validation?lang=eng) | [打印此文章](http://www.oschina.net/translate/angularjs-form-validation?print)

|  |  |  |
| --- | --- | --- |
| 今天我们将来看看Angular如何帮助做表单验证. 我们将讨论更多使用angular进行有关表单操作的东西（就像我们另外的一篇文章: [提交Ajax表单：AngularJS的方式](http://scotch.io/tutorials/javascript/submitting-ajax-forms-the-angularjs-way)). 不过不用担心，那篇文章不一定要读的.  我们将重点放在客户端验证上，并使用Angular内置的表单属性。这里有一个 [demo](http://embed.plnkr.co/lCRwhj/preview). 需求  * Name 必填 * Username 非必填，最小长度3，最大长度8 * Email 非必填，但必须是合法的email * 验证未通过的表单不能提交 * 显示一个必填或者非法email格式的错误信息 * 如果正确提交就弹出一个祝贺信息   现在知道我们的目标了吧，让我们一起来构建这个东西吧. | [leoxu](http://my.oschina.net/xuleo) [leoxu](http://my.oschina.net/xuleo) 翻译于 10个月前 1人顶  [顶](javascript:vote_block(34613);) 翻译的不错哦! | |
| Angular 的表单属性 $valid, $invalid, $pristine, $dirty Angular 提供了有关表单的属性来帮助我们验证表单. 他们给我们提供了各种有关一个表单及其输入的信息，并且应用到了表单和输入.   |  |  |  | | --- | --- | --- | | 属性类 |  | 描述 | | $valid | ng-valid | Boolean 告诉我们这一项当前基于你设定的规则是否验证通过 | | $invalid | ng-invalid | Boolean 告诉我们这一项当前基于你设定的规则是否验证未通过 | | $pristine | ng-pristine | Boolean 如果表单或者输入框没有使用则为True | | $dirty | ng-dirty | Boolean 如果表单或者输入框有使用到则为True |   Angular 也提供了有关表单及其输入框的类，以便你能够依据每一个状态设置其样式. 访问表单属性  * **方位表单:** <form name>.<angular property> * **访问一个输入框:** <form name>.<input name>.<angular property>  设置我们的表单 我们将使用一个简单的表单来做演示.  angular-form-validation  我们将需要两个文件:   * index.html 我们用来显示表单的代码 * app.js 我们的Angular应用程序和控制器 (几乎没有任何代码)  Our Form Code index.html [?](http://www.oschina.net/translate/angularjs-form-validation)   |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56 | <!-- index.html -->  <!DOCTYPE html>  <html>  <head>      <!-- CSS ===================== -->      <!-- load bootstrap -->      <link rel="stylesheet" href="http://netdna.bootstrapcdn.com/bootstrap/3.0.3/css/bootstrap.min.css">      <style>          body     { padding-top:30px; }      </style>        <!-- JS ===================== -->      <!-- load angular -->      <script src="http://code.angularjs.org/1.2.6/angular.js"></script>      <script src="app.js"></script>  </head>    <!-- apply angular app and controller to our body -->  <body ng-app="validationApp" ng-controller="mainController">  <div class="container">  <div class="col-sm-8 col-sm-offset-2">        <!-- PAGE HEADER -->      <div class="page-header"><h1>AngularJS Form Validation</h1></div>        <!-- FORM -->      <!-- pass in the variable if our form is valid or invalid -->      <form name="userForm" ng-submit="submitForm(userForm.$valid)" novalidate> <!-- novalidate prevents HTML5 validation since we will be validating ourselves -->            <!-- NAME -->          <div class="form-group">              <label>Name</label>              <input type="text" name="name" class="form-control" ng-model="name" required>          </div>            <!-- USERNAME -->          <div class="form-group">              <label>Username</label>              <input type="text" name="username" class="form-control" ng-model="user.username" ng-minlength="3" ng-maxlength="8">          </div>            <!-- EMAIL -->          <div class="form-group">              <label>Email</label>              <input type="email" name="email" class="form-control" ng-model="email">          </div>            <!-- SUBMIT BUTTON -->          <button type="submit" class="btn btn-primary">Submit</button>        </form>    </div><!-- col-sm-8 -->  </div><!-- /container -->  </body>  </html> |   这里列出了一些关键点:   * **novalidate**: 它将会组织默认的HTML5验证，因为这会由我们自己来做(我们自己的将会更棒) * 我们在输入框上应用了ng-model，以便将来自表单的数据绑定到angular变量 * username上的ng-minlength 和 ng-maxlength会自己创建其规则 * name 输入框是必填的 * email 输入框有属性 **type=”email”** | | [leoxu](http://my.oschina.net/xuleo) [leoxu](http://my.oschina.net/xuleo) 翻译于 10个月前 1人顶  [顶](javascript:vote_block(34615);) 翻译的不错哦! |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 验证规则 Angular 有很多验证规则，像是 tong-min leng than dng-max length.   Angular还可以自己配置规则.  [Angular输入](http://docs.angularjs.org/api/ng.directive:input)指引上有说明 .  [?](http://www.oschina.net/translate/angularjs-form-validation)   |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10 | <input     ng-model="{ string }"     name="{ string }"     required     ng-required="{ boolean }"     ng-minlength="{ number }"     ng-maxlength="{ number }"     ng-pattern="{ string }"     ng-change="{ string }">  </input> |   现在建好了表单, 接着创建Angular应用和控制器，ng-app ng-控制器. 应用的 Codeapp.js [?](http://www.oschina.net/translate/angularjs-form-validation)   |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | // app.js  // create angular app  var validationApp = angular.module('validationApp', []);    // create angular controller  validationApp.controller('mainController', function($scope) {        // function to submit the form after all validation has occurred      $scope.submitForm = function(isValid) {            // check to make sure the form is completely valid          if (isValid) {              alert('our form is amazing');          }        };    }); | | | [petert](http://my.oschina.net/u/1422355) [petert](http://my.oschina.net/u/1422355) 翻译于 10个月前 1人顶  [顶](javascript:vote_block(34579);) 翻译的不错哦! |
| [其它翻译版本(1)](javascript:show_other(34533,34579);) |
| 使HTML5验证器的novalidate 我们将在我们的表单使用novalidate。这是一个很好的做法，因为我们将会自己处理验证。如果我们让我们的表单这样做，它看起来很丑陋。  angular-form-novalidate 禁用提交按钮 ng-disabled 现在真正的好戏上演了。我们**开始使用Angular的属性** 。首先让我们禁用我们的提交按钮。如果我们的表单是$invalid的，我们只想禁用它。  [?](http://www.oschina.net/translate/angularjs-form-validation)   |  |  | | --- | --- | | 1  2  3  4  5  6  7 | <!-- index.html -->  ...        <!-- SUBMIT BUTTON -->      <button type="submit" class="btn btn-primary" ng-disabled="userForm.$invalid">Submit</button>    ... |   只使用一点代码（ng-disabled），如果我们的表单是$invalid的，表单按钮将被禁用。   这意味着，我们的name input 字段是必需的，并且email input字段需要一个有效的电子邮件。 | [AndyLam](http://my.oschina.net/andylam2010) [AndyLam](http://my.oschina.net/andylam2010) 翻译于 10个月前 1人顶  [顶](javascript:vote_block(34544);) 翻译的不错哦! | |
| [其它翻译版本(1)](javascript:show_other(34534,34544);) | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用 ng-show显示错误信息 **ng-valid** 和**ng-invalid** 会基于提供的表单规则自动验证输入的有效性.  咱们在输入部分加一些错误信息，只要不等于$valid或是已经使用过的就行 (不能展示还没使用).  [?](http://www.oschina.net/translate/angularjs-form-validation)   |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | <!-- index.html -->  ...        <!-- NAME -->      <div class="form-group">          <label>Name</label>          <input type="text" name="name" class="form-control" ng-model="name" required>          <p ng-show="userForm.name.$invalid && !userForm.name.$pristine" class="help-block">You name is required.</p>      </div>        <!-- USERNAME -->      <div class="form-group">          <label>Username</label>          <input type="text" name="username" class="form-control" ng-model="user.username" ng-minlength="3" ng-maxlength="8">          <p ng-show="userForm.username.$error.minlength" class="help-block">Username is too short.</p>          <p ng-show="userForm.username.$error.maxlength" class="help-block">Username is too long.</p>      </div>        <!-- EMAIL -->      <div class="form-group">          <label>Email</label>          <input type="email" name="email" class="form-control" ng-model="email">          <p ng-show="userForm.email.$invalid && !userForm.email.$pristine" class="help-block">Enter a valid email.</p>      </div>    ... |   就像这样 Angular 会根据规则来验证输入部分的$invalid 和 $pristine properties属性从而决定是否显示错误信息.  angular-form-validation-errors | | [petert](http://my.oschina.net/u/1422355) [petert](http://my.oschina.net/u/1422355) 翻译于 10个月前 1人顶  [顶](javascript:vote_block(34581);) 翻译的不错哦! |
| [其它翻译版本(1)](javascript:show_other(34535,34581);) |
| 格局类 Angular在验证输入或表单时的有效有否是已经提供了一些类，像是 (ng-valid,ng-invalid,ng-pristineandng-dirty).  你可以编辑自己喜欢的CSS . 你可以私有定制化这些类来实现特定的场景应用.  [?](http://www.oschina.net/translate/angularjs-form-validation)   |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9 | .ng-valid         {  }  .ng-invalid     {  }  .ng-pristine     {  }  .ng-dirty         {  }    /\* really specific css rules applied by angular \*/  .ng-invalid-required         {  }  .ng-invalid-minlength         {  }  .ng-valid-max-length         {  } | | [petert](http://my.oschina.net/u/1422355) [petert](http://my.oschina.net/u/1422355) 翻译于 10个月前 1人顶  [顶](javascript:vote_block(34580);) 翻译的不错哦! | |
| [其它翻译版本(1)](javascript:show_other(34536,34580);) | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 使用 ng-class 根据条件添加类 因为我们使用了 [Bootstrap](http://getbootstrap.com/), 我们将就使用它们提供的类(has-error). 这样就能围绕我们的form-group获得漂亮的错误信息和颜色.  ng-class 允许我们基于一个表达式添加类. 在这种情况下，我们想要想我们的form-group添加一个 has-error 类，如果输入框的状态是$invalid或者不是pristine的话.  其工作的方式是 ng-class="{ <class-you-want> : <expression to be evaluated > }". 更多的信息，请读一读 [Angular ngClass 文档吧](http://docs.angularjs.org/api/ng.directive:ngClass).  [?](http://www.oschina.net/translate/angularjs-form-validation)   |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | <!-- index.html -->  ...        <!-- NAME -->      <div class="form-group" ng-class="{ 'has-error' : userForm.name.$invalid && !userForm.name.$pristine }">          <label>Name</label>          <input type="text" name="name" class="form-control" ng-model="user.name" required>          <p ng-show="userForm.name.$invalid && !userForm.name.$pristine" class="help-block">You name is required.</p>      </div>        <!-- USERNAME -->      <div class="form-group" ng-class="{ 'has-error' : userForm.username.$invalid && !userForm.username.$pristine }">          <label>Username</label>          <input type="text" name="username" class="form-control" ng-model="user.username" ng-minlength="3" ng-maxlength="8">          <p ng-show="userForm.username.$error.minlength" class="help-block">Username is too short.</p>          <p ng-show="userForm.username.$error.maxlength" class="help-block">Username is too long.</p>      </div>        <!-- EMAIL -->      <div class="form-group" ng-class="{ 'has-error' : userForm.email.$invalid && !userForm.email.$pristine }">          <label>Email</label>          <input type="email" name="email" class="form-control" ng-model="user.email">          <p ng-show="userForm.email.$invalid && !userForm.email.$pristine" class="help-block">Enter a valid email.</p>      </div>    ... |   现在我们的表单就有了恰当的Bootstrap错误类.  angular-form-validation-class-errors | | [leoxu](http://my.oschina.net/xuleo) [leoxu](http://my.oschina.net/xuleo) 翻译于 10个月前 1人顶  [顶](javascript:vote_block(34626);) 翻译的不错哦! |
| 只在提交表单后显示错误信息 有时候不想在用户正在输入的时候显示错误信息. 当前错误信息会在用户输入表单时立即显示. 由于Angular很棒的数据绑定特性，这是可以发生的. 因为所有的事务都可以在一瞬间发生改变，这在表单验证时会有副作用.  对于你想要只在表单正要提交之后才显示错误消息的场景, 你就需要对上面的代码做一些小调整.   1. 你要去掉提交按钮上的ng-disabled，因为我们想要用户即使是在表单没有全部验证完的情况下也能点击提交. 2. 你要在表单已经被提交之后添加一个变量. 在你的 submitForm() 函数中, 只要加入 $scope.submitted = true 就行了;. 一旦表单被提交，它就会保存提交值为true的submitted变量. 3. 将错误规则从ng-class="{ 'has-error' : userForm.name.$invalid && !userForm.name.$pristine }" 调整为 ng-class="{ 'has-error' : userForm.name.$invalid && !userForm.name.$pristine && submitted }". 这就确保了错误消息只会在表单被提交时被显示出来. 你也许会需要为这个变量调整所有其它的 ng-class 和 ng-show.   现在，只有在submitted变量被设置为true时才会显示错误信息. | [leoxu](http://my.oschina.net/xuleo) [leoxu](http://my.oschina.net/xuleo) 翻译于 10个月前 1人顶  [顶](javascript:vote_block(34627);) 翻译的不错哦! | |

|  |  |
| --- | --- |
| 只有在输入框之外点击时才显示错误消息 只有在输入框之外点击时才显示错误消息（也被叫做失去焦点 **blur**) 这一需求比在提交时验证要复杂一点. 在失去焦点时验证表单需要一个c[ustom指令](http://docs.angularjs.org/api/ng.directive:form). 这是一个可以让你操作DOM的指令.  我们正在写一篇文章专门讨论此话题。同时，还有其他的一些资源讨论了创建custom指令来处理失去焦点的情况:   * <http://coding-issues.blogspot.in/2013/10/angularjs-blur-directive.html> * <http://blog.ejci.net/2013/08/06/dealing-with-focus-and-blur-in-angularjs-directives/> * <http://plnkr.co/edit/Xfr7X6JXPhY9lFL3hnOw?p=preview>  全部完成 现在一旦我们正确填写了所有的信息，我们的表单提交按钮就能使用了. 一旦我们提交了表单，我们将会见到我们设置的弹出消息.  angular-form-validation-alert  用了几行我们就可以:   * 进行输入框验证 * 显示表单错误消息 * 定制样式类 * 自动禁止和启用表单 * 定制规则   如你所见，我们使用了Angular内置的表单验证技术来创建一个客户端验证表单. | [leoxu](http://my.oschina.net/xuleo) [leoxu](http://my.oschina.net/xuleo) 翻译于 10个月前 1人顶  [顶](javascript:vote_block(34628);) 翻译的不错哦! |
| 未来 照现在的情况，在用户在输入框之外点击 时做验证不是一个简单的过程. Angular团建已经意识到这个问题，他们已经宣称计划要添加更多的状态来处理 像 form.submitted,input.$visited, input.$blurred, or input.$touched的情况. 这里有一些有关未来的表单验证的资源:   * [Github 问题](https://github.com/angular/angular.js/issues/583#issuecomment-31277556) * [ngForm 模块设想](https://docs.google.com/a/scotch.io/document/d/192dCUnoIBQ7-vurvPeS9BElGdxfk0Ddcof2KzDDi1Mc/edit)   希望尽快能有轻松做表单验证的方案出来，用以处理我们的应用程序的不同状态.  **编辑 #1**: 添加有关在表单提交或者失去焦点之后进行验证的信息. 还添加了一些资源. **编辑 #2**: 修改表单函数的处理过程来获得valid参数. 对于创建可测试控制器的帮助. 这一点要感谢 [Fredrik Bostrom](https://twitter.com/fredrikbostrom).      本文是[**AngularJS 表单**](http://scotch.io/series/angularjs-forms)**系列的一部分.**   * 提交[AJAX表单: AngularJS的方式](http://scotch.io/tutorials/javascript/submitting-ajax-forms-the-angularjs-way) * AngularJS表单验证(本文) * [在Angular表单中处理复选框和单选框](http://scotch.io/tutorials/javascript/handling-checkboxes-and-radio-buttons-in-angular-forms)   查看代码 : <http://plnkr.co/edit/lCRwhj?p=preview> | | |

# [Angularjs使用指令做表单校验](http://segmentfault.com/a/1190000002391494)

## 前言

通常，使用angular做表单验证，一般都是把验证规则单独写为service，然后通过依赖注入的方式调用。在个别情况下，例如用户注册表单，需要根据用户输入给出不同提示信息，使用service略显不合适宜，所以可以采用指令的方式。

## 简易表单

如下为一个简易表单，有四项提示信息，依据状态呈现。校验规则为数字，大写字母，小写字母至少出现两项，通过正则配合ng-pattern能够实现相同的效果，此处仅为引入指令校验，不做深究。其中，user-validator为自定义指令。

<form class="form-horizontal" name="form" role="form" ng-controller="ValidCtrl">

<div class="form-group">

<label for="user" class="col-sm-2 control-label" >用户名</label>

<div class="col-sm-5">

<input type="text" ng-model="user" name="user" id="user" user-validator class="form-control" required="required">

</div>

<label class="col-sm-5" ng-show="form.user.$pristine">请输入用户名</label>

<label class="col-sm-5" ng-show="form.user.$error.required && form.user.$dirty">

用户名不能为空

</label>

<label class="col-sm-5" ng-show="form.user.$error.defined && form.user.$dirty">

用户名不符合规则

</label>

<label class="col-sm-5" ng-show="form.user.$valid && form.user.$dirty">

用户名有效

</label>

</div>

</form>

## 校验指令

校验指令代码如下：

angular.module('shuffleApp', [])

.directive('userValidator', ['$log', function($log) {

return {

restrict: 'A',

require: 'ngModel',

link: function($scope, $element, $attrs, $ngModelCtrl) {

var verifyRule = [/^\d+$/, /^[a-z]+$/, /^[A-Z]+$/];

var verify = function(input) {

return !(verifyRule[0].test(input) ||

verifyRule[1].test(input) ||

verifyRule[2].test(input));

};

$ngModelCtrl.$parsers.push(function(input) {

var validity = verify(input);

$ngModelCtrl.$setValidity('defined', validity);

return validity ? input : false;

});

$ngModelCtrl.$formatters.push(function(input) {

var validity = verify(input);

$ngModelCtrl.$setValidity('defined', validity);

return validity ? input : false;

})

}

}

}]);

指令内容非常简单，检测是否全为数字，小写字母，大写字母，然后取反即得到校验结果。然后在$parser, $formatter内部设置校验结果即可。