Angular表单

1. 模板式表单：需要引入FormsModule’@angular/forms’

表单的数据模型是通过组件模板中的相关指令来定义的，因为使用这种方式定义表单的数据模型时，我们会受限于HTML的语法，所以，模板驱动方式只适合用于一些简单的场景。

1. 响应式表单:需要引入ReactiveFormsModule’@angular/forms’

使用响应式表单时，你通过编写TypeScript代码而不是html代码来创建一个底层的数据模型，在这个模型定义好以后，你使用一些特定的指令，将木板上的html元素与底层的数据模型连接在一起。

1. 模板式表单中的指令：都来自于FormsModule

NgFrom: ----》FormGroup

自动的被添加到form标签上

(1)可以在form标签之外使用：<div ngform>

(2)当你不希望使用的时候，需要明确的指出：

<form ngNoForm>

(3)可以被模板本地变量引用

<form #myForm=”ngForm”>

<div>

{{myForm.value | json}}

</div>

(4)会拦截标准的html的提交事件，组织表单提交

用一个ngSubmit的事件来替代它

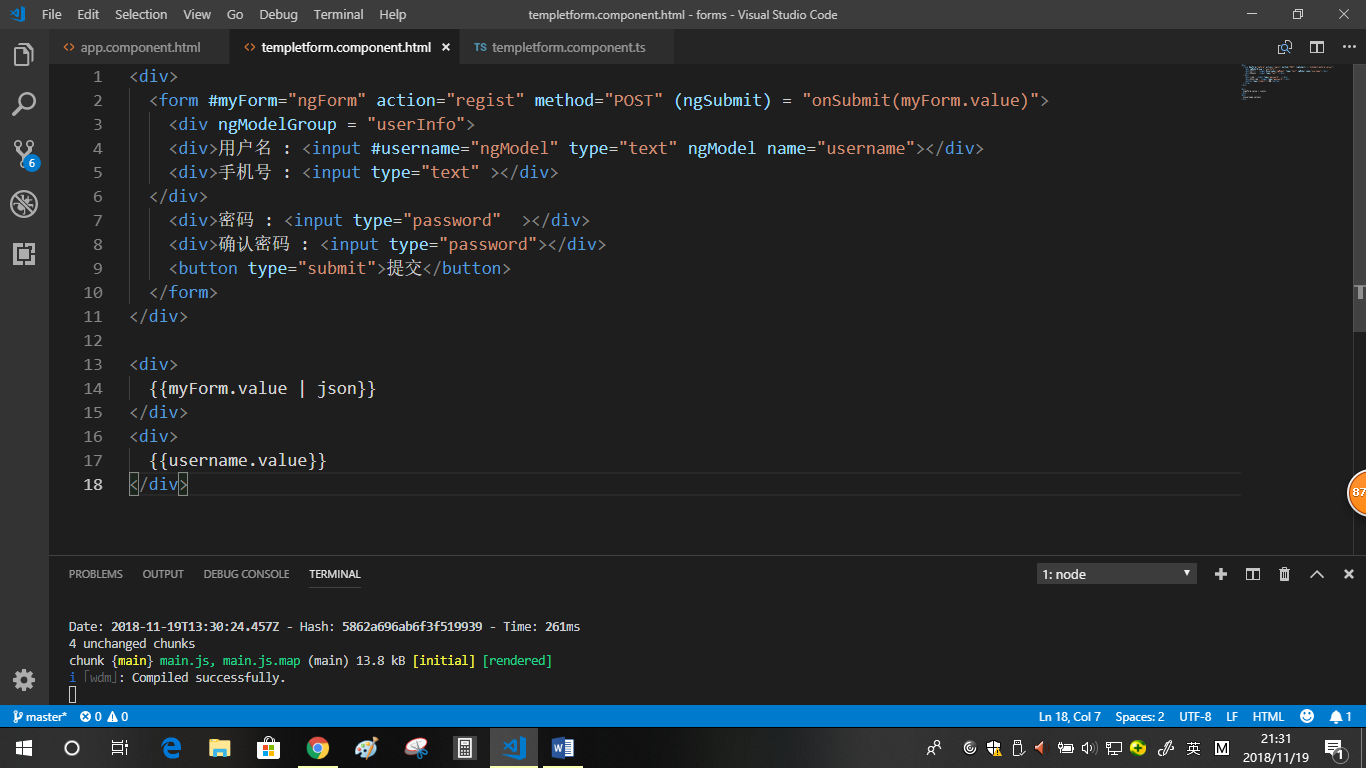
<form #myForm=”ngForm” （ngSubmit）=”onSubmit(myForm.value)”>

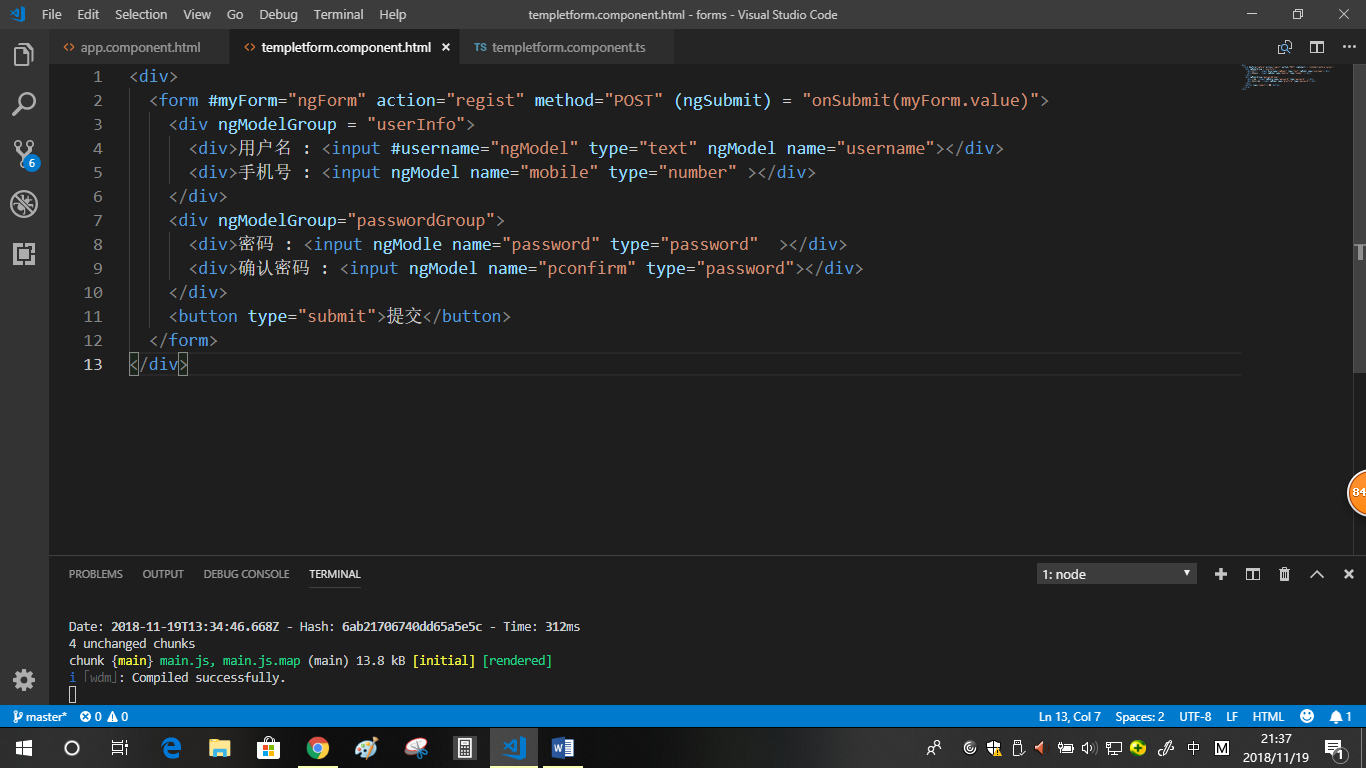
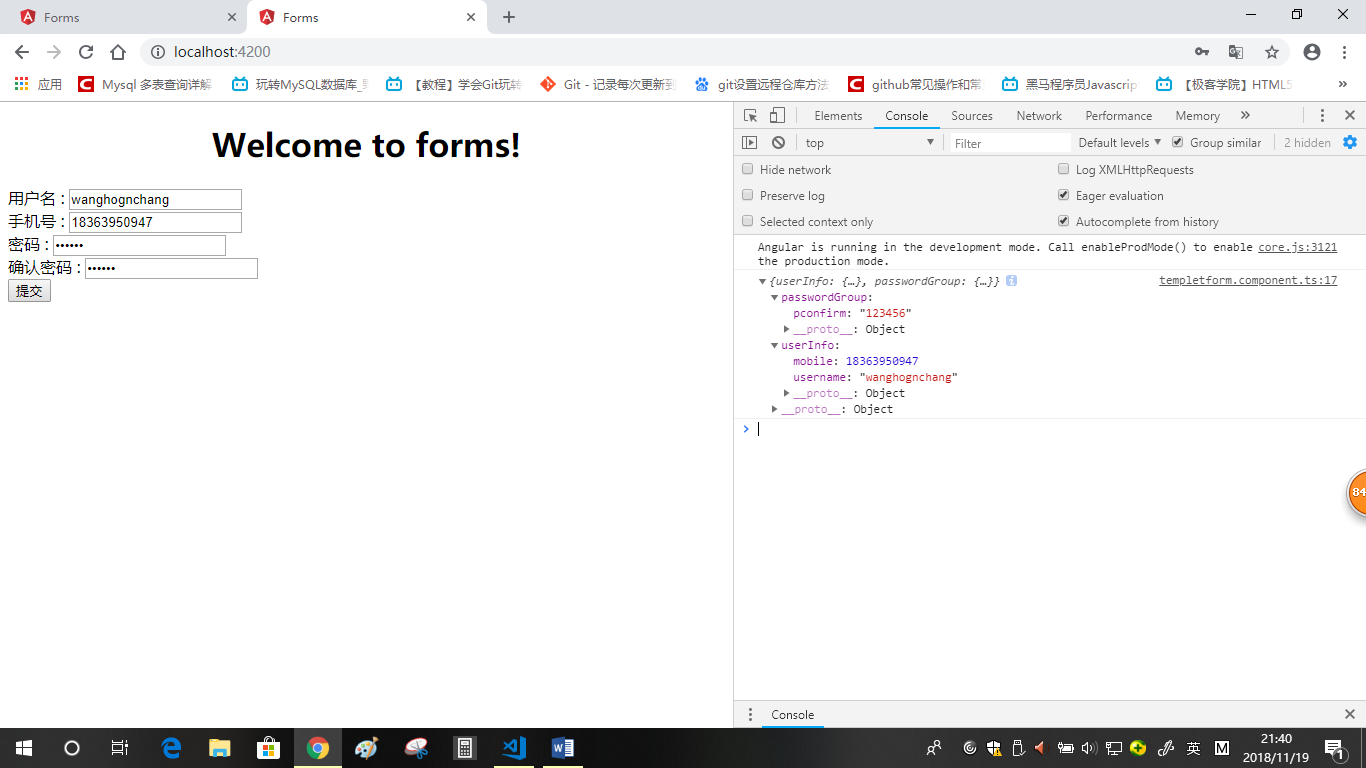
NgModel----🡪FormControl

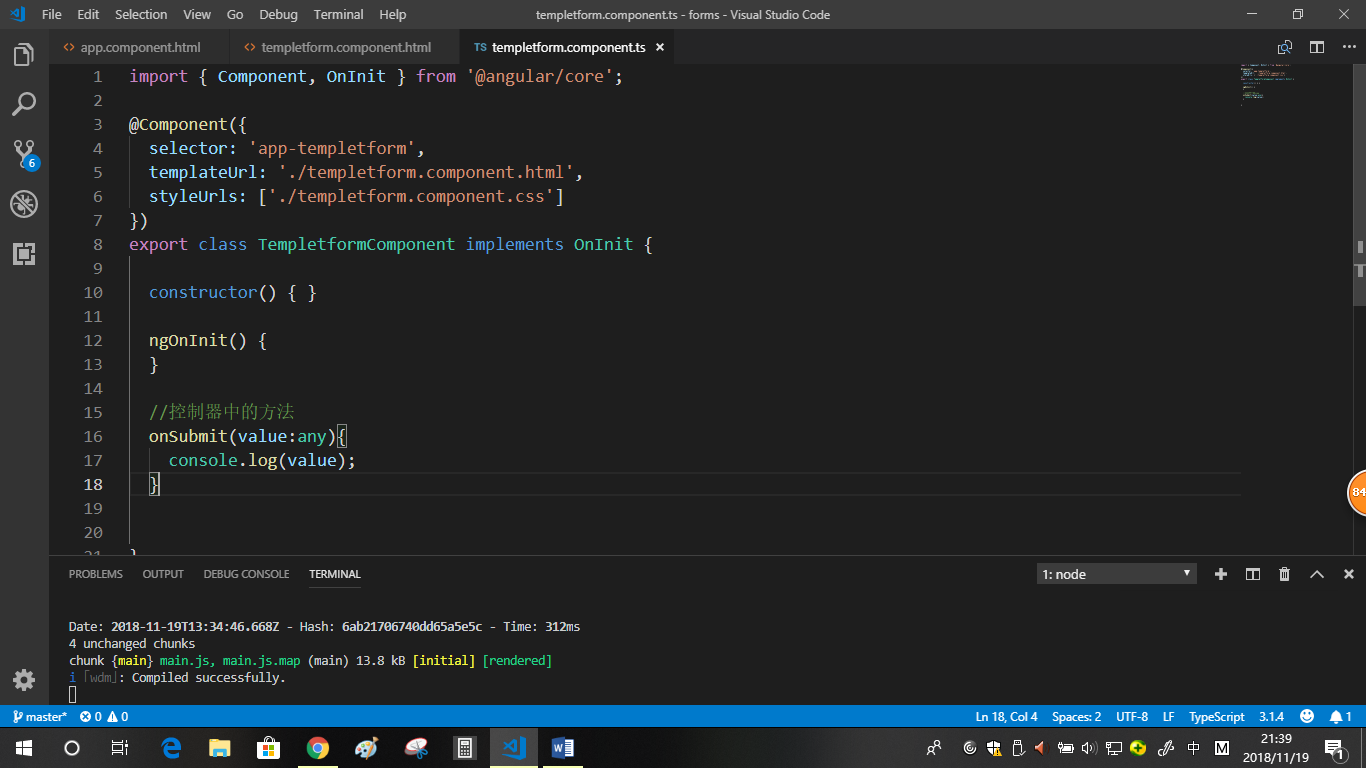
不需要用小括号括起来直接写一个ngModel就行了，不过需要添加一个name属性

NgModelGroup：----》FormGroup

代表表单的一部分



模板式表单重构之后的结果：



1. 相应式表单：需要引入ReactiveFormsModule

分两步进行创建：

1. 编码创建数据模型
2. 通过指令将模板元素绑定到数据模型上
3. 三个类：

FormControl

Username:FormControl = new FormControl(“aaa”);

FormGroup

formModel : FormGroup = new FormGroup({

from:new FormControl(),

to:new FormControl()

});

FormArray:有一个长度属性，可变长度的

Emails : FormArray = new FormArray([

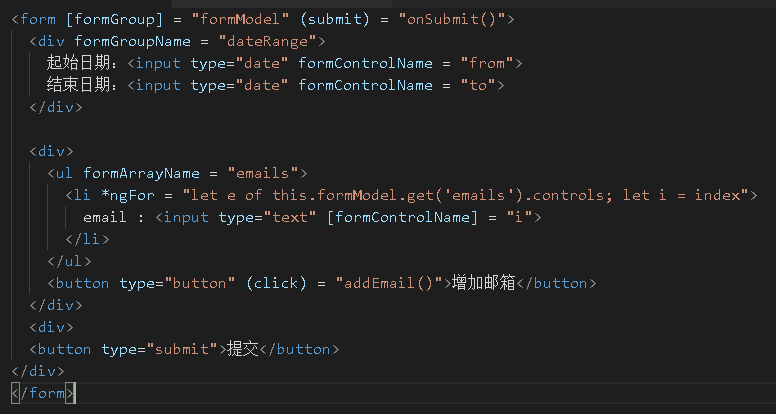
new FormControl(“aaa”),

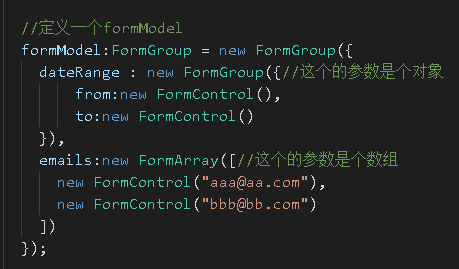
new FormControl(“bbb”)

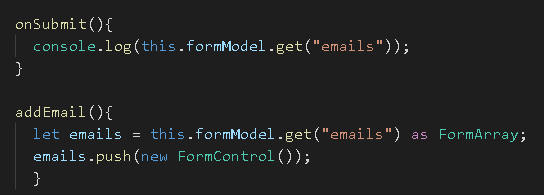
]);

(2)响应式表单的指令

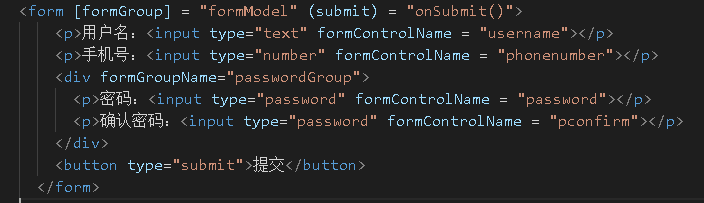
|  |  |  |  |
| --- | --- | --- | --- |
| 类 | | 指令 | |
| FormControl | | formControl | formControlName |
| FormGroup | | formGroup | formGroupName |
| FormArray |  | formArrayName | |

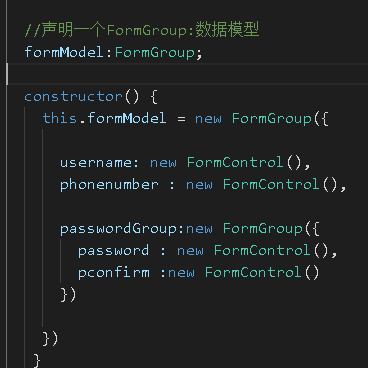


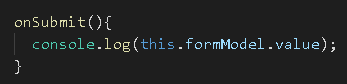




上面的三个图都是重点，反复看几遍！！！

下面是重构的注册的表单：

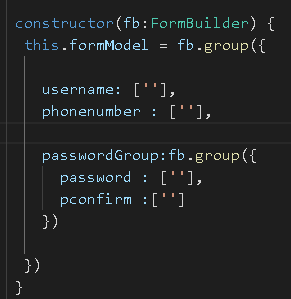




一定要注意整个表单的写法：创建数据模型，表单模板的写法

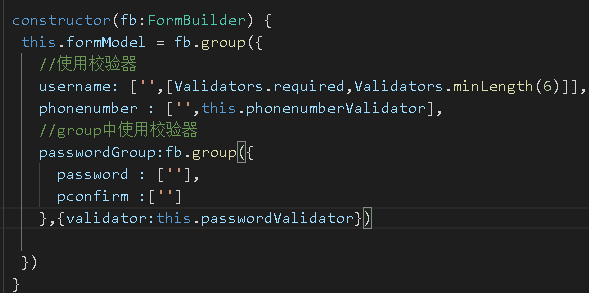
都非常的关键

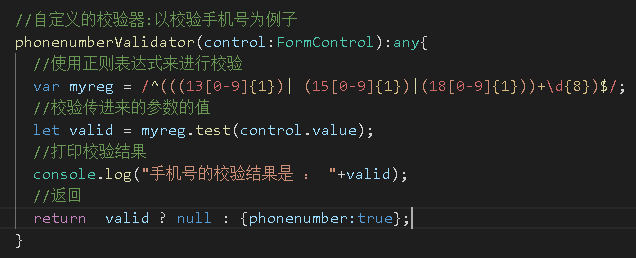
使用FormBuilder进行数据模型的创建：

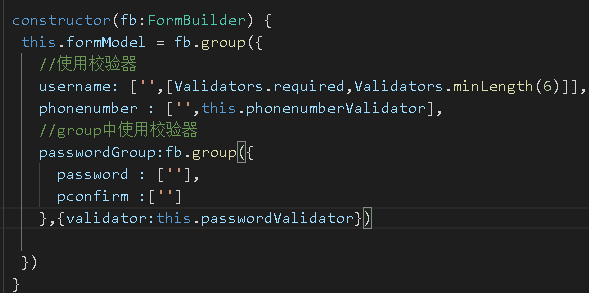


1. 表单校验
2. 校验器：就是一个普通的方法，用来验证数据格式是否正确的
3. Angular中有已经定义好的校验器，如Validators类
   1. 使用预定义的校验器：

直接写就行了，使用Validators类的相关方法



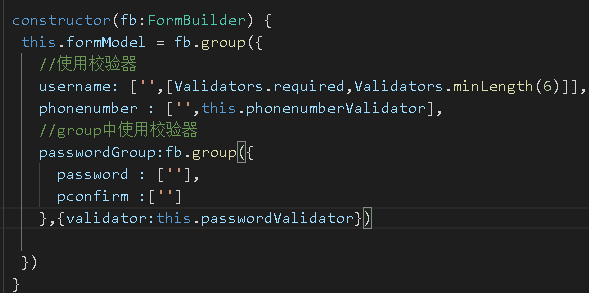
（2）在FormControl中使用校验器：



在formControl的后面添加上校验器就可以了

（3）在FormGroup中使用校验器：

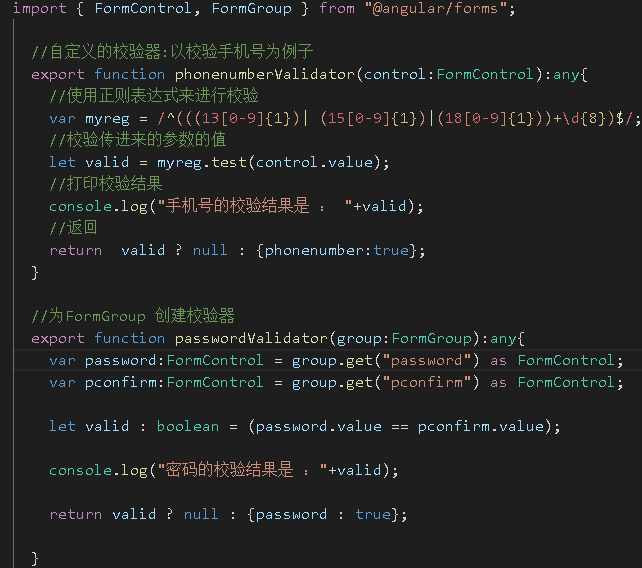


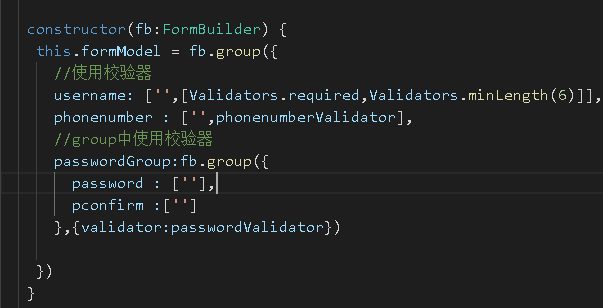


这个是不一样的，需要在formGroup中写一个对象来使用！

以上的校验器都是直接写在控制器中的，当然也可以单独拿出来写在一个ts文件中，这样就可以使这个校验器能够得到重用！

(4)将校验器放入到一个单独的ts文件中，然后通过export暴露出来，这样使其他的组件都可以得到重用这些校验器

调用时候的操作：

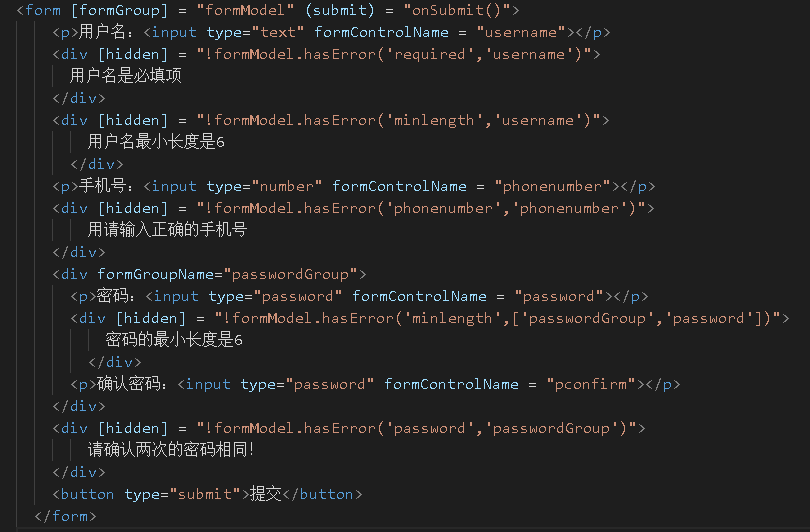


在使用的时候需要进行引用，不过当路径找不到时，直接放到根目录下面

当所有的字段都加上校验器之后，就可以对整个的formModel的数据模型进行校验，当所有的字段校验都通过了之后，formModel的才会通过。

This.formModel.valid

(6)当出现错误时候，在模板上面提示错误信息给用户：

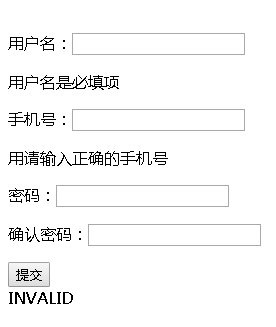
 (6.1)硬编码的方式给出提示：

第二个参数是要校验的字段

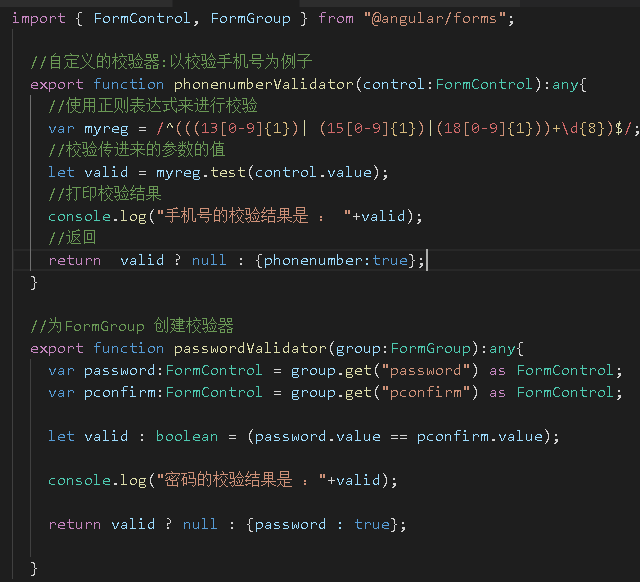
第一个参数是校验器的返回的名字

hasError方法的使用，控制hidden属性，控制是否显示

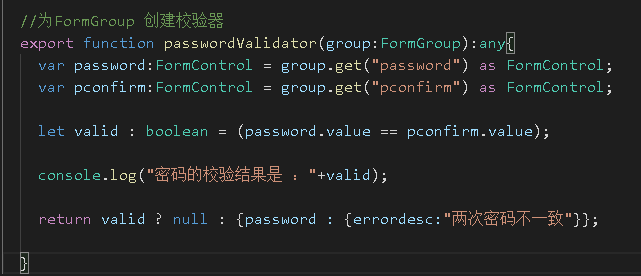
有一个不一样的内容：当包含在另一个校验器当中时，需要使用数组的形式进行参数的传递



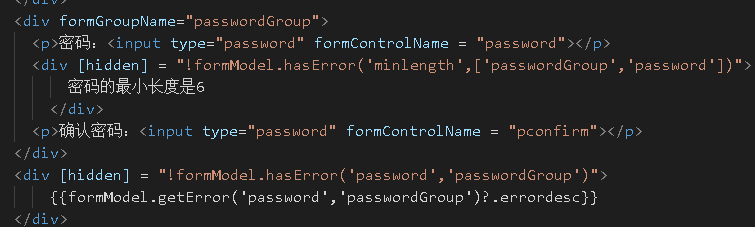
实际上刚打开的时候是不应该显示这些的，在后面的时候会处理这些



Phonenumber和password就是上面的第一个参数

(6.2)通过校验器的返回值提示错误的信息：

在校验器中写好要提示的内容



这里使用的方法换了，该车给你了getError

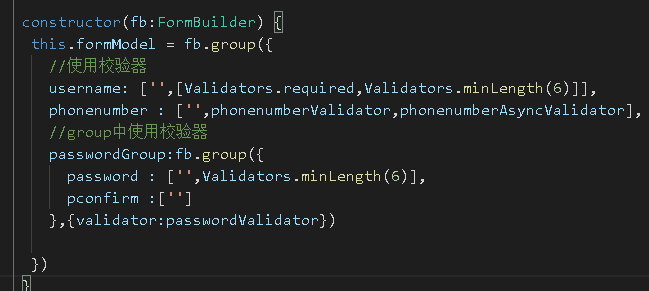
在这里显示错误的提示信息，注意写法

(7)异步校验器：可以通过调用远程的服务来检查值，异步校验器返回的是一个可被观测的流。



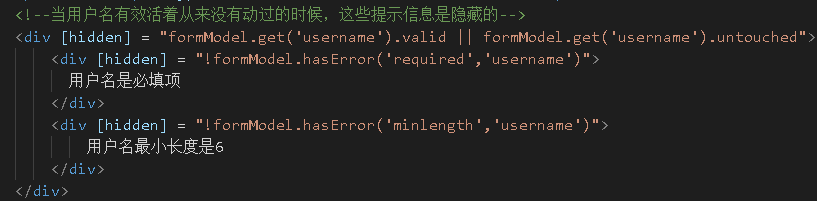


第三个参数出现了：就是异步的校验器

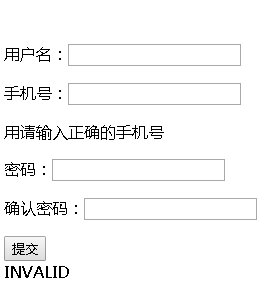


(8)状态字段：

Touched 和 untouched :是否获取过焦点

一般来控制错误信息是否来显示

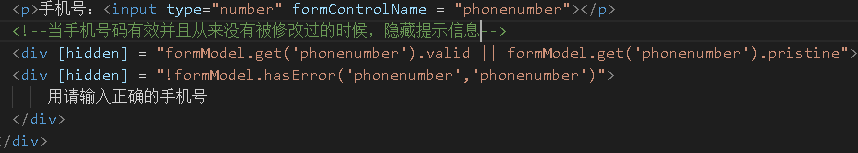
当username有效活着从来没有被动过的时候，隐藏错误提示信息



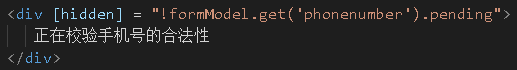
刚出现页面的时候是没有提示信息的，解决了上面的问题

Pristine 和 dirty

值从来没有被改变过：pristine=true,dirty=false



Pending：当一个字段为异步校验时，为pending状态



(9)为其添加相关的样式：

例如，当用户名无效并且被碰过了之后，边框变红

第五章中css样式绑定的内容

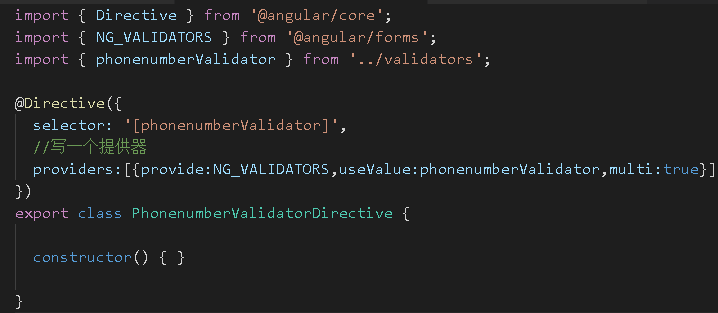
(10) 模板式表单的校验：因为控制其中是没有数据模型的，所以只能使用指令进行操作，将校验的方法封装成指令。

(10.1)创建指令：

ng g directive directives/phonenumberValidator

ng g directive directives/passwordValidator

例如下图：



这个属性为true说明前面的provide是可以重复使用的，也就是每一个校验器都写那个名字

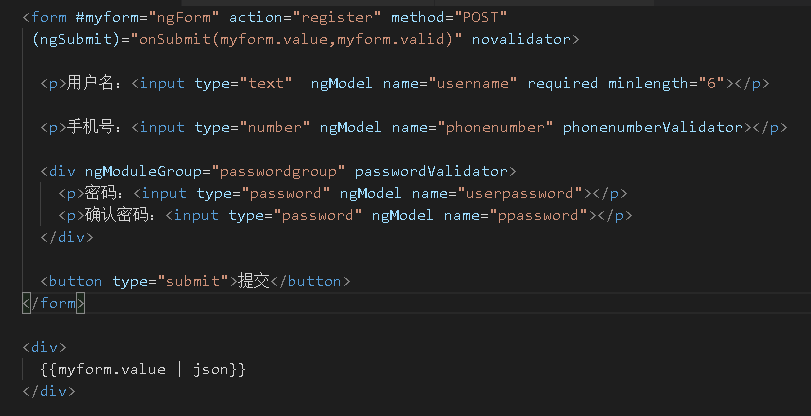
这个方法就是你自己写的校验的方法

这个NG\_VALIDATORS是固定的

这个名字可以改，就是要使用时候的指令，并且是个中括号，说明是当作属性来用的

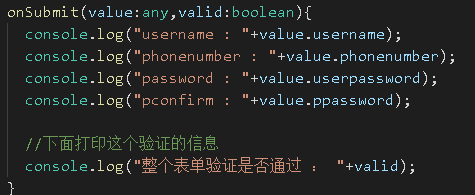
(10.2)指令与组建的区别：指令没有模板文件，其他的没有区别。

声明关闭浏览器的自带校验，这样required之类的才是Angular中的校验

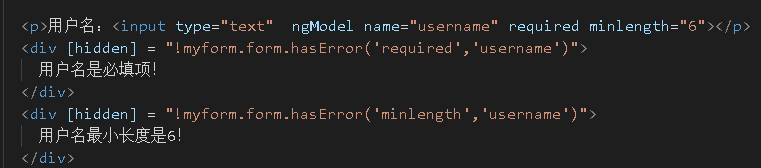
(10.3)指令的使用：

因为控制其中没有数据模板，所以只能通过参数传递回去

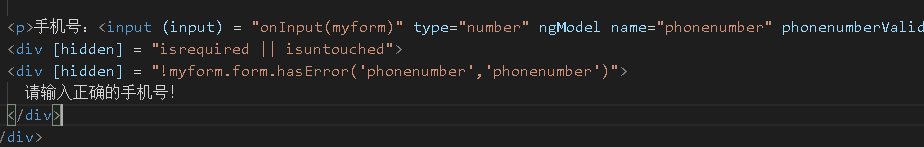
使用校验器

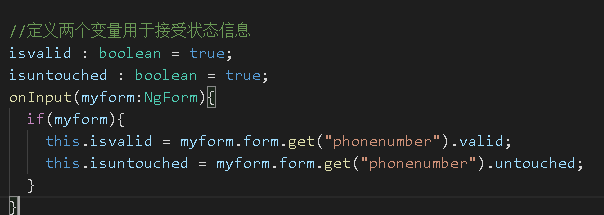


在这里接收和打印传过来的信息

(10.4)控制错误信息的显示：仍然是使用hasError方法，只不过是使用模板中的引用来进行调用这个方法的

这个要写成这样了

(10.5)通过状态属性控制什么时候显示的问题



上面的这个实验思路是这样的，但是没有成功，我不知道为什么，后面可以再试一下！