# Angular 表单

# 表单基础

## 表单的介绍

表单使用场景非常广泛，常见的场景有用户注册、用户登录、数据的添加修改、问卷调查、文件上传等。

在一个应用系统做WEB框架技术选型时如何处理表单，应该是最重要的考虑因素之一，在我们大多数场景的项目下，核心功能无非就是增删改查（CRUD），而表单在增加和修改中起到非常重要的作用。

## 表单功能

一般情况下表单做这三件事：

1、显示表单项。

2、校验用户输入值。

3、提交表单数据。

## 纯HTML表单

HTML内置了表单标签，但它的一些标签特性存在浏览器兼容问题，并且自定义校验规则、表单数据获取、处理、提交等流程比较复杂。

表单应该有5个业务上的功能：

1. 每一个输入字段都可以独立的指定校验规则。
2. 如果用户输入不符合规则，我们需要提示校验信息。
3. 彼此依赖的字段需要被一起校验。（密码/确认密码。邮箱/确认邮箱）
4. 应用应该可以控制提交到服务器的表单的值。（处理数据）
5. 应用可以控制数据如何提交（HTTP/ajax/ ws）

纯HTML表单1/2 能够满足，3/4/5不可以。

## Angular的表单

在Angular中表单处理超越了常规的数据绑定，将表单处理作为整个框架的顶级特性。在表单中数据模型是指用来保存表单数据的数据结构。不管是那种表单，都有一个对应的数据模型来存储表单的数据。数据模型并不是一个任意的对象，它是一个由angular/forms模块中的一些特定的类如：FormControl，FormGroup，FormArray等组成的。在模板式表单中，不能访问这些类。需要注意的是在响应式表单中同样需要自己来编写模板。

表单中的三种数据模型他们分别是三种类：

FormControl：构成表单的基本单位（例如：一个input等）。

FormGroup：构成表单的顶级、构成表单分组，是FormControl的集合。

FormArray：代表了可以持续增长或减少的数据集合（可变的），管理了任意多个内容，跟FormGroup相比它的内容中没有key这个属性，只能通过下标来访问内容。

# 模板式表单

模板驱动(Template-Driven Forms)称之为模板式表单。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、模板驱动方式只适用于一些简单的场景。（AngularJS中就是模板式表单）在模板式表单中，数据模型是由angular基于组件模板中的指令隐式创建的。

需要在根模块中引入FormsModule来使用模板式表单的相关API。

Angular中表单的特点：当我们的form标签在Angular项目中启动渲染的时候， Angular就会默认的给Form标签加上一个 ngForm的指令。

模板式表单中的三种指令：

1. ngForm 指令就代表了整个表单，会自动添加到form标签上，创建一个FormGroup的顶层实例。（刚才场景下“注册”按钮的点击事件就被抓取屏蔽了）
2. ngModel 这个指令能处理双向数据绑定，在表单中标记当前HTML元素为表单中的一部分，创建一个FormControl数据模型。
3. ngModelGroup 可以对表单输入内容进行分组，方便我们在语义上区分不同类型的输入。

注意：模板式表单的控制文件（.ts）中不会使用任何表单相关的API。

# 响应式表单

模型驱动(Model-Driven Forms)响应式表单。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。在响应式表单中，我们通过编码明确的创建数据模型然后将模板上的html元素与底层的数据模型连接在一起。

需要在根模块中引入ReactiveFormsModule来使用模板式表单的相关API。

创建响应式表单的步骤：

1. 编码创建数据模型。
2. 将数据模型连接到html中。

响应式表单中的三种指令：

1. 要绑定FormControl： [formControl]=”定义的变量” / formControlName=” 定义的变量名称”
2. 要绑定FormGroup： [formGroup]=”” / formGroupName=””
3. 要绑定 FormArray： formArrayName=””

**上述指令顶层的FormGroup指令使用[formGroup] 方式来绑定，而内部绑定的情况下只能使用 formGroupName / formControlName。**

响应式表单的提交方法是：（submit）/ 模板式表单：(ngSubmit)

# 表单校验

## 内置验证规则

内置验证大全：<https://www.angular.cn/api/forms/Validators>

在模板中查看校验：formModel.hasError方法有两个参数。第一需要校验的规则key；第二需要校验的字段名称。

## 自定义校验

所有我们的数据模型类，例如FormControl、FormGroup都是继承了我们的一个父类叫作：AbstractControl 。

API地址：<https://www.angular.cn/api/forms/AbstractControl>

步骤：

1. 新建一个普通的ts文件。
2. 新建一个用export修饰的方法。
3. 方法依赖的参数需要一个AbstractControl的子类。默认会传递当前所附着在的那个元素。
4. 该方法需要返回null或者一个[key: string]: any 类型的返回值。（null | mobile: true）

注意：自定义的校验通过使用formModel.hasError 来获取的结果时候，第一个参数传递的是校验返回失败时那个 key 的值！

异步校验：参见mobileAsyncValidator方法，以及它的使用。

多控件一起校验： 参见equalValidator 方法，以及使用。这时校验器使用时调用formModel.hasError方法的时候第二个参数要加上所属group的前缀值例如：

['passwordsGroup','password']。

### 校验器

自定义的方法 xxx(control: AbstractControl): {[key: string]: any} {}

### 响应式表单校验

参见reactive-form 组件。

### 模板式表单校验

只能使用指令的方式来实现校验。例如直接在html元素上写 required

# 状态字段

当表单初始化的时候，用户并没有操作过。这时显示校验的错误信息会显得很不友好。所以Angular提供了状态字段来处理这种场景。

## 状态字段

包括4组分别为：

1. touched和untouched。判断控件是否别访问过。（控件是否获取过焦点）
2. dirty和pristine。判断控件值是否被改变过。（值被修改过dirty为true）
3. pending。控件正处于异步校验时pending为true。
4. valid和invalid。判断控件值的校验是否通过。

## 表单控件状态的css类

包括如下几种：

.ng-valid

.ng-invalid

.ng-pending

.ng-pristine

.ng-dirty

.ng-untouched

.ng-touched

分别对应处于相应状态的字段。当表单元素处于对应的状态时Angular 会自动把控件属性作为 CSS 类映射到控件所在的元素上。

# 实际开发中的响应式表单

参见reactive-form-with-bootstrap 组件中的内容。

# 表单控件全家福

参见 all-type 组件中的内容

# 补充

## 表单的change事件监听

通过this.formModel.valueChanges 方法来监听输入的改变。

## 表单赋值

通过patchValue 或者 setValue 两个方法都可以。区别在于patchValue可以传多于我们数据模型的内容。

# 作业

1. 自学ControlValueAccessor 接口，它可以把 form 模型中的值映射到视图中，也在视图发生变化时，通知 form directives 或 form controls。我们在第五节课“组件（下）”中完成的内容可以通过实现（implements）ControlValueAccessor，来让我们自定义的控件使用在模板式和响应式表单中。
2. 自学动态表单：<https://www.angular.cn/guide/dynamic-form>

作业1的难度较大，作业2相对而言简单一些。同学们根据实际情况自行选择。