**disable**

https://docs.rxweb.io/decorators/disable/decorators

**禁用控制条件**

它用于有条件地禁用控件

**什么时候使用**

我们使用验证器检查了conditionalExpression，但现在出现了一种情况，根据条件，我们需要启用或禁用控件。作为一个标准的角度接近，我们将[禁用]和它的条件。一旦条件成功，它将禁用控件。

这里，我们将通过@disable装饰器管理被禁用的控件。我们必须设置条件表达式。

让我们看看如何有条件地禁用控件。

**Basic disable decorator**

场景:在响应式表单中，我们有两个字段modeType和fullName。当用户点击此时的Change Mode按钮时，它将禁用或启用基于modeType = view或modeType = edit的控件。

首先，我们需要创建一个用户模型，并在模型中定义一个modeType和fullName属性，以实现功能需求。

export class User{

@prop() modeType:string;

@disable({conditionalExpression:function(control:AbstractControl){return this.modeType == "View"; }})

@prop() fullName:string;

}

通过Angular的FormBuilder服务，我们在组件中创建了FormGroup。

export class DisableAddComponent implements OnInit {

currentMode: string = "View";

userFormGroup: FormGroup;

constructor(private formBuilder: RxFormBuilder) { }

ngOnInit() {

let user = new User();

user.modeType = this.currentMode;

this.userFormGroup = this.formBuilder.formGroup(user);

}

changeMode() {

this.currentMode = this.currentMode == "View" ? "Edit" : "View";

this.userFormGroup.controls.modeType.setValue(this.currentMode);

}

}

接下来，我们需要编写html代码。用于基于按钮单击和启用或禁用控件的切换条件

<form \*ngIf="userFormGroup" [formGroup]="userFormGroup">

<div class="form-group">

<button class="btn btn-primary" \*ngIf="userFormGroup.value.modeType == 'View'" (click)="changeMode()">

Change Mode Edit</button>

<button class="btn btn-primary" \*ngIf="userFormGroup.value.modeType == 'Edit'" (click)="changeMode()">

Change Mode View</button>

</div>

<div class="form-group">

<label>Full Name</label>

<input type="text" formControlName="fullName" class="form-control" />

</div>

</form>

**elementClass**

https://docs.rxweb.io/decorators/elementClass/decorators

有条件地将类应用于表单控件元素

您可以通过TS代码有条件地设置元素类

**Basic elementClass decorator**

让我们创建一个用户模型，并在模型中定义密码属性。

export class User {

@prop()

@elementClass({

conditionalExpression: function (c) {

return this.password == "complex" && c.dirty == true ? 'ng-invalid' : '';

}})

userName: string;

@prop()

password: string;

}

你必须使用RxFormBuilder服务在组件中创建FormGroup。

**error**

https://docs.rxweb.io/decorators/error/decorators

**什么时候使用**

在UI上有条件地显示错误消息有点笨拙。作为一种标准的angular方法，我们使用\*ngIf条件并写入错误消息。一旦条件成功，它将显示错误消息。

这里，我们将通过@error装饰器管理错误消息。我们必须设置条件表达式。

让我们看看如何有条件地显示错误消息。

**Basic error decorator**

场景:在响应式表单中，有两个字段firstName和userName。主要的需求是提交没有被禁用，当用户点击提交时，它将显示无效表单字段的错误消息。

要在应用程序中全局配置消息，请参考ReactiveFormConfig

让我们创建一个用户模型，并在模型中定义freeText的属性。

export class FormField{

@prop() action:string;

@error({conditionalExpression:function(control:AbstractControl){ return this.action === "submit"}})

@required()

firstName:string;

@error({conditionalExpression:function(control:AbstractControl){ return this.action == "submit"}})

@required()

userName:string;

}

你必须使用RxFormBuilder服务在组件中创建FormGroup。

export class ErrorAddComponent implements OnInit {

errorObject = {}

userInfoFormGroup: FormGroup

constructor(

private formBuilder: RxFormBuilder

) { }

ngOnInit() {

var formField = new FormField();

this.userInfoFormGroup =<RxFormGroup>this.formBuilder.formGroup(formField);

}

submit(){

if(this.userInfoFormGroup.invalid)

this.userInfoFormGroup.controls.action.setValue("submit");

}

}

接下来，我们需要编写html代码。为了显示错误信息，你必须使用“errorMessage”属性。

<form [formGroup]="userInfoFormGroup">

<div class="form-group">

<label>First Name</label>

<input type="text" formControlName="firstName" class="form-control" />

<small class="form-text text-danger" >{{userInfoFormGroup.controls.firstName["errorMessage"]}}</small>

</div>

<div class="form-group">

<label>User Name</label>

<input type="text" formControlName="userName" class="form-control" />

<small class="form-text text-danger" >{{userInfoFormGroup.controls.userName["errorMessage"]}}</small>

</div>

<button (click)="submit()" class="btn btn-primary">Submit</button>

</form>

**model**

https://docs.rxweb.io/decorators/model/decorators

**@model**

这对于同样的验证以及有条件地显示错误消息很有帮助。

**Example**

让我们创建一个用户模型，并在模型中定义密码属性。

@model([{ propNames: [":all:"], validationConfig: { required: true } }])

export class User{

@prop() userName:string;

@prop() password:string;

@prop() cityName:string;

@prop() countryName:string;

@prop() areaName:string;

}

你必须使用RxFormBuilder服务在组件中创建FormGroup。

**elementClass**

当您希望有条件地对多个属性应用elementClass时。

@model([{

propNames: [":all:"],

elementClass: {

conditionalExpression: function (c) {

return this.password == "complex" && c.dirty == true ? 'dirty' : '';

}}

}])

export class User{}

**error**

当你想在两个以上的属性上设置error属性时，在模型装饰器的propsConfig中设置error属性会更容易。

@model([{

propNames: [":all:"],

validationConfig: { required: true },

error: {conditionalExpression:function(control:AbstractControl){ return this.userName == "john"} }

}, ])

export class User{}

**排除属性动态**

当你有两个以上的属性时，在模型装饰器的propsConfig中设置ignore属性会更容易。

@model([{

propNames: ["userName"],

ignore:(x)=>x.type == "Admin"}])

export class User{}

**prop**

https://docs.rxweb.io/decorators/prop/decorators

它用于定义模型类中字段的属性。

假设您想要创建一个用户表单，其中包含像emailAddress这样的字段，并且您想要在该字段上使用道prop饰器。

**Basic prop decorator**

首先，我们需要创建一个用户模型，并在模型中定义一个emailAddress属性，以实现功能需求。

export class User {

@prop() emailAddress: string;

}

通过Angular的FormBuilder服务，我们在组件中创建了FormGroup。

export class PropAddComponent implements OnInit {

userFormGroup: FormGroup

constructor( private formBuilder: RxFormBuilder) { }

ngOnInit() {

let user = new User();

this.userFormGroup = this.formBuilder.formGroup(user);

}

}

**Binding服务器端属性名**

如果客户端属性名和服务器属性名不同。然后将服务器属性名绑定到客户端。

使用prop绑定服务器端名称

export class User {

@prop({name:'email\_Address'})

emailAddress: string;

}

export class PropServerComponent implements OnInit {

userFormGroup: FormGroup

constructor(private formBuilder: RxFormBuilder) { }

ngOnInit() {

let user = new User();

let formBuilderConfiguration = new FormBuilderConfiguration();

let userData = {

email\_Address:'bharat.patel@gmail.com',

};

formBuilderConfiguration.dynamicValidation = JSON.parse(JSON.stringify(userData));

this.userFormGroup = this.formBuilder.formGroup(user,formBuilderConfiguration);

}

}

**设置默认值**

当你想设置属性的默认值时，你必须在创建FormGroup的多个组件中设置默认值，而不是使用一个有效的方法来设置默认值，它应该通过decorator在属性上应用。

export class User {

@prop({defaultValue:'bharat.patel@gmail.com'})

emailAddress:string

}

**排除条件属性**

当需要根据特定条件排除属性时，必须在propConfig中添加ignore属性。

export class User {

@prop()

type:string;

@prop({ignore:function(){ return this.type == "Personal";}})

emailAddress:string;

}

**isPrimaryKey**

它被用来定义模型的主键，你需要设置isPrimaryKey属性为true。

export class User {

@prop({ isPrimaryKey: true})

userId: number;

@prop()

firstName: string;

}

**propArray**

https://docs.rxweb.io/decorators/propArray/decorators

propArray用于定义一个类的实例数组。

假设您想要创建一个User表单，其中包含User类中的CourseName等字段，

这是一个类Course的实例数组，如果您想添加@required验证，那么可以使用@propArray()。

**Basic PropArray Decorator**

首先，我们需要创建一个Course类，并在其中创建一个类型为字符串的属性courseName。创建一个User类，并使用@propArray()装饰器在其中创建一个Array类型的课程属性。

export class Address {

@prop() city: string;

@prop() country: string;

}

export class User {

@prop()

emailAddress: string;

@propArray(Address)

addresses: Address[];

}

通过Angular的FormBuilder服务，我们在组件中创建了FormGroup。

export class PropArrayAddComponent implements OnInit {

ngOnInit() {

let user = new User();

user.addresses = new Array<Address>();

let address = new Address();

user.addresses.push(address);

this.userFormGroup = this.formBuilder.formGroup(user);

}

addAddress(){

let addresses = this.userFormGroup.controls.addresses as FormArray;

addresses.push(this.formBuilder.formGroup(Address));

}

onSubmit(){

}

}

**PropArray based on Entity Instance**

如果您有条件地想要propArray实体的实体。

如果contactType属性值等于address，那么address模型属性应该包含在contact的嵌套表单组中，或者属性值包含telephone，那么telephone模型属性应该包含在嵌套表单组中。

export class Address {

@prop() city: string;

@prop() country: string;

}

export class Telephone {

@prop() mobileNumber: number;

}

export class User {

@prop()

contactType: string;

@propArray(undefined, { entityProvider: function () { return this.contactType == "Address" ? Address : Telephone } })

contacts: Telephone[] | Address[]

}

export class PropArrayEntityComponent implements OnInit {

contactType: string = "Address";

userFormGroup: FormGroup;

constructor(private formBuilder: RxFormBuilder) { }

ngOnInit() {

let user = new User();

user.contactType = "Address";

user.contacts = new Array<Address>();

user.contacts = new Array<Telephone>();

this.userFormGroup = this.formBuilder.formGroup(user);

}

addAddress() {

let addresses = this.userFormGroup.controls.contacts as FormArray;

addresses.push(this.formBuilder.formGroup(Address));

}

addTelephone() {

let telephones = this.userFormGroup.controls.contacts as FormArray;

telephones.push(this.formBuilder.formGroup(Telephone));

}

changeValue(value: string) {

this.userFormGroup.reset();

let user = new User();

this.userFormGroup.controls.contactType.setValue(value);

user.contactType = value;

this.contactType = value;

if (this.userFormGroup.controls.contactType.value == "Address") {

user.contacts = new Array<Address>();

let address = new Address();

user.contacts.push(address);

this.userFormGroup = this.formBuilder.formGroup(user);

}

else {

user.contacts = new Array<Telephone>();

let contact = new Telephone();

user.contacts.push(contact);

this.userFormGroup = this.formBuilder.formGroup(user);

}

}

}

**绑定propArray上的服务器端属性名**

如果客户端属性名和服务器属性名不同。然后将propArray的服务器属性名绑定到客户端。

export class Address {

@prop() city: string;

@prop() country: string;

}

export class User {

@prop({name:'email\_address'})

emailAddress: string;

@propArray(Address,{name:"user\_address"})

addresses: Address[];

}

**PropArray基于allowMaxIndex**

当您想要验证在proprArray中添加的索引数时，可以使用它。您需要在allowMaxIndex属性中指定它。

export class Address {

@prop() city: string;

@prop() country: string;

}

export class User {

@prop() emailAddress: string;

@propArray(Address ,{ allowMaxIndex: 2 })

addresses: Address[];

}

**创建BlankFormArray**

它用于创建一个初始化的formArray。您需要指定createBlank值为true。

export class Address {

@prop() city: string;

@prop() country: string;

}

export class User {

@prop() emailAddress: string;

@propArray(Address,{createBlank:true})

addresses: Address[];

}

**propObject**

https://docs.rxweb.io/decorators/propObject/decorators

它用于定义模型类中字段属性的对象。

假设您想要创建一个用户表单，其中包含诸如emailAddress、城市和国家等字段。您需要创建地址类的一个对象，并在用户模型类中访问它的属性。

**Basic propObject decorator**

首先，我们需要创建一个用户模型并在用户类中定义一个emailAddress属性，在用户模型中创建Address类并在其中定义城市和国家属性。在用户类中创建地址类的propoject，实现功能需求。

export class Address {

@prop() city: string;

@prop() country: string;

}

export class User {

@prop() emailAddress: string;

@propObject(Address)

address: Address;

}

通过Angular的FormBuilder服务，我们在组件中创建了FormGroup。

export class PropObjectAddComponent implements OnInit {

userFormGroup: FormGroup

constructor( private formBuilder: RxFormBuilder ) { }

ngOnInit() {

let user = new User();

user.address = new Address();

this.userFormGroup = this.formBuilder.formGroup(user);

}

}