rxweb/reactive-form-validators 使用指南

<https://docs.rxweb.io/getting-started>

**allOfAsync**

https://docs.rxweb.io/async-validators/allOfAsync/validators  
allOfAsync验证验证器将检查用户是否已输入给定字段的所有值或非异步。

**什么时候使用**假设您想要创建一个雇员表单，其中包含部门、projectDomains、资格、技能和爱好等字段。您希望用户输入特定字段的所有值。在这里，该字段以数组的形式获取，并根据此选择通过异步提供matchValues应用于属性。在这里，根据需求可能会出现这些场景。

在projectDomains字段上应用allOfAsync验证，您希望用户选择matchValues中提到的所有项目域。  
在兴趣爱好字段上显示自定义验证消息。  
使用messageKey属性显示语言字段上的自定义消息。  
基于服务器规则动态应用allOfAsync验证。

让我们看看allOfAsync验证器是如何满足需求的。

**基本allOfAsync验证**通过Angular的FormBuilder服务，我们在组件中创建了FormGroup。  
这里我们介绍了添加表单操作。

@Component({

selector: 'app-allOfAsync-add-validator',

templateUrl: './all-of-async-add.component.html'

})

export class AllOfAsyncAddValidatorComponent implements OnInit {

employeeInfoFormGroup: FormGroup

projectDomains: string[] = [];

projectDomainsArray: string[] = ["ECommerce", "Banking", "Educational", "Gaming"];

constructor(

private formBuilder: FormBuilder, private http: HttpClient) { }

ngOnInit() {

this.employeeInfoFormGroup = this.formBuilder.group({

department: [''],

projectDomains: ['',[], RxwebValidators.allOfAsync({validatorConfig:of({ matchValues: ["ECommerce", "Banking"] })})],

});

}

addProjectDomain(element: any, index: number) {

var indexOf = this.projectDomains.indexOf(element.value);

element.checked ? this.projectDomains.push(element.value) : this.projectDomains.splice(indexOf, 1);

this.employeeInfoFormGroup.controls.projectDomains.setValue(this.projectDomains);

}

}

**ValidatorConfig**

类型:Observable  
Validator配置用于根据服务器规则使用关键字的动态解析属性的值，如下所示  
下面的选项在RxwebValidators.allOfAsync(({validatorConfig:of({<property\_name>:<value>})}))验证器中不是强制使用的。如果需要，那么使用下面的选项。

Option Description

matchValues： matchValues是一个数组，用于验证值的匹配。根据它，数组中的所有值都应该匹配

message： 覆盖全局配置消息并在各自的FormControl上设置自定义错误消息

messageKey： 当用户希望在其某些字段上显示不同的自定义验证消息时，可以使用BaseConfig的messageKey属性。用户可以通过在ReactiveFormConfig中定义一个全局自定义messageKey，并在验证的message属性中设置它。

**matchValues**

Type : any[]

RxwebValidators.allOfAsync({validatorConfig:of({ matchValues: ["ECommerce", "Banking"] })

**message**

Type : string

RxwebValidators.allOfAsync(

{validatorConfig:

of({matchValues:["Drawing", "Singing","Dancing","Travelling"],message: "请选择所有的爱好"}

)

**messageKey**

Type : string

RxwebValidators.allOfAsync(

{validatorConfig:of({ matchValues: ["English", "Spanish", "French"], messageKey: 'allOfAsyncMessageKey' })

**完整的allOfAsync示例**这个完整的allOfAsync示例包含了所有的ArrayConfig属性，它将满足场景1、2、3和4的要求。

ngOnInit() {

this.employeeInfoFormGroup = this.formBuilder.group({

department: [''],

projectDomains: ['', [], RxwebValidators.allOfAsync({ validatorConfig: of({ matchValues: ['ECommerce', 'Banking', 'Educational', 'Gaming'] }) })],

qualifications: ['', [], RxwebValidators.allOfAsync({

validatorConfig: of({

matchValues: ['Secondary', 'Senior Secondary', 'B.Tech.', 'M.Tech.', 'B.C.A.', 'M.C.A.'],

conditionalExpression: (x, y) => x.department == 'DotNet'

})

})],

skills: ['', [], RxwebValidators.allOfAsync({

validatorConfig: of({

matchValues: ['MVC', 'AngularJS', 'Angular 5', 'C#', 'Web Api', 'SQL Server'],

conditionalExpression: 'x => x.department ==\'DotNet\''

})

})],

hobbies: ['', [], RxwebValidators.allOfAsync({

validatorConfig: of({

matchValues: ['Drawing', 'Singing', 'Dancing', 'Travelling', 'Sports'],

message: 'Please select all hobbies'

})

})],

languages: ['', [], RxwebValidators.allOfAsync({

validatorConfig: of({

matchValues: ['English', 'Spanish', 'French'],

messageKey: 'allOfAsyncMessageKey'

})

})]

});

}

**choiceAsync**

<https://docs.rxweb.io/async-validators/choiceAsync/validators>

choiceAsync验证验证器将检查用户输入的值是否与属性中定义的范围异步匹配

**什么时候使用**

假设您想要创建一个雇员表单，其中包含部门、projectDomains、资格、技能和爱好等字段。您希望用户为特定的choiceAsync选择特定范围的值。这里以数组的形式获取字段，并根据此情况，通过异步应用minlength和maxLength将choiceAsync应用于属性。在这里，根据需求可能会出现这些场景。

projectDomains字段，您希望用户在其中输入最少三个域。  
您希望用户最多输入4个条件的条件字段。  
在爱好字段中添加自定义消息。  
使用messageKey属性显示certification字段上的自定义消息。  
基于服务器规则动态应用choiceAsync验证。

languages: ['', [], RxwebValidators.choiceAsync({

validatorConfig: of({

minLength: 2,

conditionalExpression: (x, y) => x.department == 'DotNet'

})

})],

skills: ['', [], RxwebValidators.choiceAsync({

validatorConfig: of({

minLength: 2,

conditionalExpression: 'x => x.department ==\'DotNet\''

})

})],

hobbies: ['', [], RxwebValidators.choiceAsync({

validatorConfig: of({

maxLength: 4,

message: 'Please select upto 4 hobby'

})

})],

certifications: ['', [], RxwebValidators.choiceAsync({

validatorConfig: of({

minLength: 2,

messageKey: 'choiceMessageKey'

})

})],

noneOfAsync  
noneOfAsync验证验证器将检查是否没有从给定输入中异步选择任何值。

<https://docs.rxweb.io/async-validators/noneOfAsync/validators>

**什么时候使用**让我们假设您正在创建一个雇员表单，您希望雇员在其中不输入任何包含部门、爱好和技能等字段的特定值。在这里，该字段以数组的形式获取，并通过异步应用matchvalues将选择应用于属性。在这里，根据需求可能会出现这些场景。

您希望用户根据匹配值在其中输入技能的技能字段。  
应用noneOfAsync验证基于匹配条件的表单，如如果部门是' dotnet '，那么技能值应该基于匹配值。  
爱好上的自定义消息字段。  
使用messageKey属性显示语言字段上的自定义消息。  
基于服务器规则动态应用noneOfAsync验证。

this.employeeInfoFormGroup = this.formBuilder.group({

department: [''],

projectDomains: ['', [], RxwebValidators.noneOfAsync({ validatorConfig: of({ matchValues: ['ECommerce', 'Banking', 'Educational', 'Gaming'] }) })],

qualifications: ['', [], RxwebValidators.noneOfAsync({

validatorConfig: of({

matchValues: ['Secondary', 'Senior Secondary', 'B.Tech', 'M.Tech', 'B.C.A.', 'M.C.A.'],

conditionalExpression: (x, y) => x.department == 'DotNet'

})

})],

skills: ['', [], RxwebValidators.noneOfAsync({

validatorConfig: of({

matchValues: ['MVC', 'AngularJS', 'Angular 5', 'C#', 'Web Api', 'SQL Server'],

conditionalExpression: 'x => x.department ==\'DotNet\''

})

})],

hobbies: ['', [], RxwebValidators.noneOfAsync({

validatorConfig: of({

matchValues: ['Drawing', 'Singing', 'Dancing', 'Travelling', 'Sports'],

message: 'Please do not select any hobby'

})

})],

languages: ['', [], RxwebValidators.noneOfAsync({

validatorConfig: of({

matchValues: ['English', 'Spanish', 'French'],

messageKey: 'noneOfMessageKey'

})

})]

});

**oneOfAsync**  
oneOfAsync验证验证器将检查用户是否异步输入了任何一个给定的输入。

<https://docs.rxweb.io/async-validators/oneOfAsync/validators>

**什么时候使用**让我们假设您正在创建一个雇员表单，您希望雇员在其中输入某个特定值的任何一个值，该值包含部门、爱好和技能等字段。在这里，该字段以数组的形式获取，并根据此，oneOfAsync通过应用matchvalues异步应用于属性。在这里，根据需求可能会出现这些场景。

您希望用户根据匹配值在其中输入技能的技能字段。  
根据表单中的匹配条件应用oneOfAsync验证，比如如果部门是“dotnet”，那么技能值应该基于匹配值。  
爱好上的自定义消息字段。  
使用messageKey属性显示语言字段上的自定义消息。  
基于服务器规则动态应用oneOfAsync验证。

this.employeeInfoFormGroup = this.formBuilder.group({

department: [''],

projectDomains: ['',[],

RxwebValidators.oneOfAsync({ matchValues: ["ECommerce", "Banking", "Educational", "Gaming"] })})],

qualifications: ['',[],

RxwebValidators.oneOfAsync({ matchValues:

["Secondary", "Senior Secondary", "B.Tech", "M.Tech", "B.C.A.", "M.C.A."],

conditionalExpression: (x, y) => x.department == 'DotNet' })})

],

skills: ['',[],

RxwebValidators.oneOfAsync({ matchValues:

["MVC", "AngularJS", "Angular 5", "C#", "Web Api", "SQL Server"],

conditionalExpression: "x => x.department =='DotNet'" })})],

hobbies: ['',[],

RxwebValidators.oneOfAsync({ matchValues:

["Drawing", "Singing", "Dancing", "Travelling", "Sports"],

message: "Please select atleast 1 hobby" })})],

languages: ['',[],

RxwebValidators.oneOfAsync({ matchValues:

["English", "Spanish", "French"],

messageKey: 'oneOfAsyncMessageKey' })})]

});

**patternAsync**patternAsync验证验证器将允许用户异步输入匹配预定义的patternAsync值参数的输入。

<https://docs.rxweb.io/async-validators/patternAsync/validators>

允许输入只包含字母的用户名。  
根据表单中匹配的条件应用patternAsync验证，例如，如果Username是John，那么只有patternAsync验证必须应用到Age值(即:Age)。，年龄字段只能是数字)。  
在Zipcode字段上添加自定义消息。  
使用messageKey属性显示Pin Code字段上的自定义消息。  
基于服务器规则动态应用patternAsync验证。

this.userFormGroup = this.formBuilder.group({

userName: ['', [], RxwebValidators.patternAsync({ validatorConfig: of({ expression: { 'onlyAlpha': /^[A-Za-z]+$/ } }) })],

zipCode: ['', [], RxwebValidators.patternAsync({

validatorConfig: of({

expression: { 'zipCode': /^[0-9]{5}(?:-[0-9]{4})?$/ },

message: 'Zip code should match 12345 or 12345-6789'

})

})],

pinCode: ['', [], RxwebValidators.patternAsync({

validatorConfig: of({

expression: { 'pinCode': /^[1-9][0-9]{5}$/ },

messageKey: 'patternMessageKey'

})

})]

});