Angular FormArray Example

25 Comments / 5 minutes of reading / March 28, 2024

← Form Group | Angular Tutorial | Nested/Dynamic Form Array →

The Angular FormArray example shows how to use the `FormArray`. The FormArray allows us to add controls dynamically to the reactive forms. In this example, we will take a very simple task of dynamically adding/removing skills to an employee form.

Table of Contents

What is FormArray

The `FormArray` is a way to manage the collection of Form Controls in Angular. The controls can be a `FormGroup`, `FormControl`, or another `FormArray`.

We can group Form Controls in Angular forms in two ways. One is using the `FormGroup` and the other one is `FormArray`. The difference is how they implement it. In `FormGroup` controls becomes a property of the `FormGroup`. Each control is represented as key-value pair. While in `FormArray`, the controls become part of an array

Because it is implemented as an Array, it makes it easier dynamically add controls.

FormArray Example

Let us build a simple app, which allows us to add the new skill of a person dynamically.

Import FormArray

To use FormArray, First, you need to import the `FormArray` from the Angular Forms Module.

```
1
2 import { FormGroup, FormControl,FormArray, FormBuilder } from '@angular/forms'
3
```

Build a Form Model

Build a form model `skillsForm` using the FormBuilder. Our Form has two fields. `name` of the person and his `skills`. Since the person can have more than one skill, we define `skills` as `FormArray`.

```
skillsForm: FormGroup;

constructor(private fb:FormBuilder) {

  this.skillsForm = this.fb.group({
    name: '',
    skills: this.fb.array([]) ,
  });

}
```

Next, a `getter` method `skills`, which returns the `skills` FormArray from the `skillsForm`

```
get skills() : FormArray {
  return this.skillsForm.get("skills") as FormArray
}
```

The skill FormGroup

We need to capture two fields under each skill. Name of the `skill` & years of `exp`. Hence we create a `FormGroup` with two fields. The method `newSkill` creates a new `FormGroup` and returns it. Note that we won't be able to assign a name to Form Group.

```
newSkill(): FormGroup {
   return this.fb.group({
     skill: '',
     exp: '',
   })
}
```

Dynamically adding skill

Now, we need to add a new skill to the `skills` FormArray. Since it is an `array` we can use the `push` method to add the new skill using the the `newSkill` method. Note that `newSkill()` method returns a `FormGroup`. The name of the `FormGroup` is its Index in the `FormArray`.

```
1
2 addSkills() {
3   this.skills.push(this.newSkill());
4 }
5
```

Dynamically Removing Skill

Use the `removeAt` method to remove the element from the `skills` FromArray.

```
1
2 removeSkill(i:number) {
3  this.skills.removeAt(i);
4 }
5
```

Submit

```
1
2  onSubmit() {
3    console.log(this.skillsForm.value);
4  }
5
```

Template

Now, it is time to build the Template. Use the `[formGroup]="skillsForm"` to bind the form to the `skillsForm` model. The `formControlName="name"` directive binds the `name` input element to `name` property of the `skillsForm`

```
1
```

```
<form [formGroup]="skillsForm" (ngSubmit)="onSubmit()">

  <p>
    <label for="name">Name </label>
    <input type="text" id="name" name="name" formControlName="name">
  </p>

 <p>
    <button type="submit">Submit</button>
  </p>

</form>

```

Binding FormArray to Template

We use the `formArrayName` directive to bind the `skills` form array to the `div` element. Now the `div` and anything inside the `div` element is bound to the `skills` form array.

```

<div formArrayName="skills">

</div>

```

Inside the `div` use `ngFor` to loop through each element of `skills` FormArray. `let i=index` will store the `index` value of the array in template local variable `i`. We will make use of it to remove the element from the `skills` array.

```

<div formArrayName="skills">
  <div *ngFor="let skill of skills().controls; let i=index">

  </div>
</div>

```

Each element under the `skills` is a `FormGroup`. We do not have a name to the `FormGroup`. The Index of the element is automatically assigned as the name for the

element.

Hence we use the `[formGroupName]="i"` where `i` is the index of the FormArray to bind the `FormGroup` to the `div` element.

```
<div formArrayName="skills">
  <div *ngFor="let skill of skills().controls; let i=index">
     <div [formGroupName]="i">

     </div>
  </div>
</div>
```

Finally, we add the controls using the `formControlName` directive.

```
Skills:
<div formArrayName="skills">
  <div *ngFor="let skill of skills().controls; let i=index">
    <div [formGroupName]="i">
      {{i}}
      skill name :
      <input type="text" formControlName="skill">
      exp:
      <input type="text" formControlName="exp">

      <button (click)="removeSkill(i)">Remove</button>
    </div>
  </div>
</div>
```

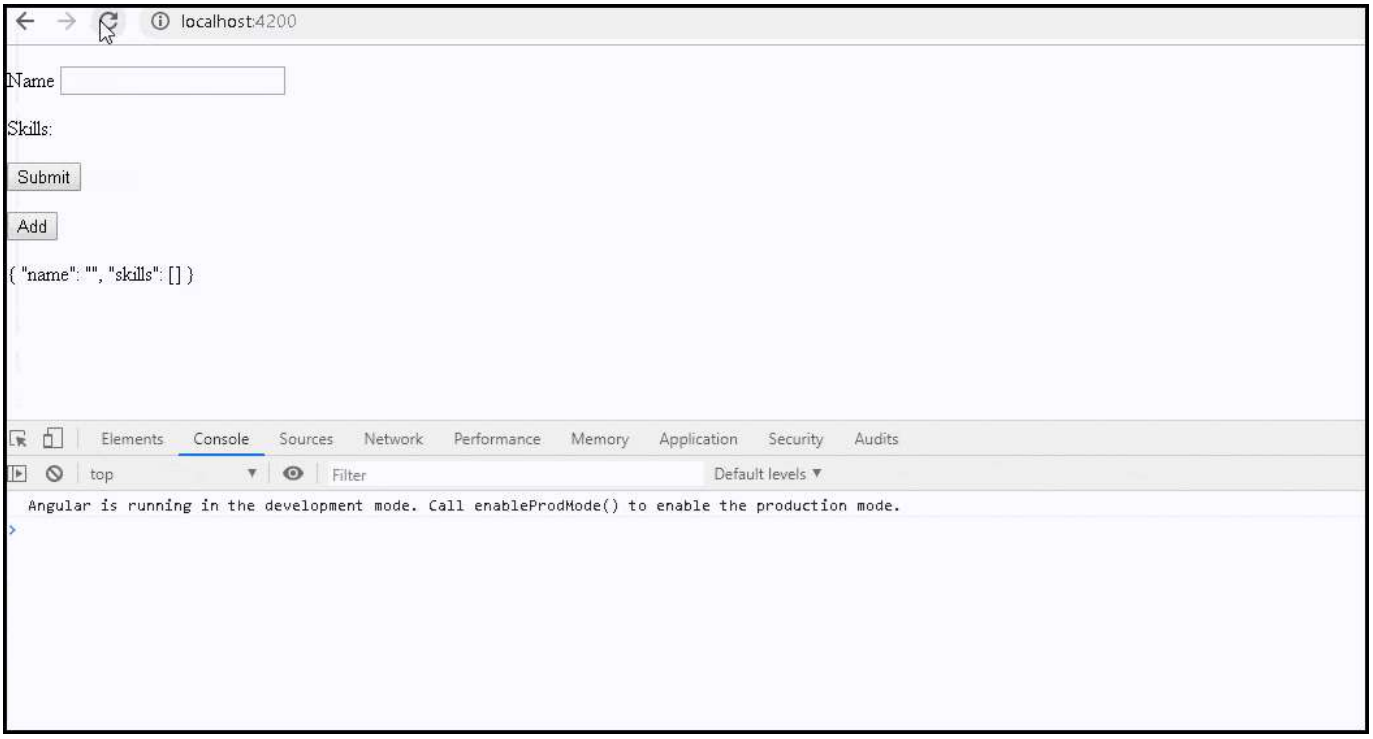
Also, pass the index `i` to `removeSkill`

```
<button (click)="removeSkill(i)">Remove</button>
```

Finally, call the `addSkills` method to add new skills.

```
<p>
  <button type="button" (click)="addSkills()">Add</button>
</p>
```

That's it



Angular FormArray Example App Running

Source Code

app.component.ts

```
import { Component, ViewChild, ElementRef } from '@angular/core';
import { FormGroup, FormControl,FormArray, FormBuilder } from '@angular/forms'


@Component({
  selector: 'app-root',
```

```
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent  {

  title = 'FormArray Example in Angular Reactive forms';

  skillsForm: FormGroup;

  constructor(private fb:FormBuilder) {

    this.skillsForm = this.fb.group({
      name: '',
      skills: this.fb.array([]) ,
    });

  }

  get skills() : FormArray {
    return this.skillsForm.get("skills") as FormArray
  }

  newSkill(): FormGroup {
    return this.fb.group({
      skill: '',
      exp: '',
    })
  }

  addSkills() {
    this.skills.push(this.newSkill());
  }

  removeSkill(i:number) {
    this.skills.removeAt(i);
  }

  onSubmit() {
    console.log(this.skillsForm.value);
  }

}


export class country {
  id: string;
  name: string;

  constructor(id: string, name: string) {
```

```
    this.id = id;
    this.name = name;
  }
}

```

app.component.html

```

<form [formGroup]="skillsForm" (ngSubmit)="onSubmit()">

  <p>
    <label for="name">Name </label>
    <input type="text" id="name" name="name" formControlName="name">
  </p>


  Skills:
  <div formArrayName="skills">
    <div *ngFor="let skill of skills().controls; let i=index">
      <div [formGroupName]="i">
        {{i}}
        skill name :
        <input type="text" formControlName="skill">
        exp:
        <input type="text" formControlName="exp">

        <button (click)="removeSkill(i)">Remove</button>
      </div>
    </div>
  </div>

  <p>
    <button type="submit">Submit</button>
  </p>

</form>


<p>
  <button type="button" (click)="addSkills()">Add</button>
</p>

{{this.skillsForm.value | json}}

```

References

1. FormArray
2. FromArrayName

Summary

In this tutorial, we learned how to create a simple FormArray Example app.

Complete List of Articles on Angular Forms

1. Angular Forms Tutorial: Fundamental & Concepts
2. Template Driven Forms in Angular
3. Set Value in Template Driven forms in Angular
4. Reactive Forms in Angular
5. FormBuilder in Reactive Forms
6. SetValue & PatchValue in Angular
7. StatusChanges in Angular Forms
8. ValueChanges in Angular Forms
9. FormControl
10. FormGroup
11. FormArray Example
12. Build Dynamic or Nested Forms using FormArray
13. Validations in Reactive Forms in Angular
14. Custom Validator in Reactive Forms
15. Passing Parameter to Custom Validator in Reactive Forms
16. Inject Service into Custom Validator
17. Validation in Template Driven Forms
18. Custom Validator in Template Driven Forms

← Form Group Angular Tutorial