Nested FormArray Example Add Form Fields Dynamically

14 Comments / 5 minutes of reading / March 9, 2023

← FormArray Example
Angular Tutorial
Setvalue & PatchValue FormArray →

In this guide, we will show you how to build a multi-level nested FormArray Example. The only way to build nested forms in angular is by using the `FormArray`. We show you how to add form fields dynamically in a 2 level nested Form. Our Form will consist of an employee and his skills. The user will be able to add/remove employee's and under each employee, you can add/remove any number of skills. If you have not used FormArray, then we suggest you read the FormArray Example in Angular.

Table of Contents

Angular Nested Formarray

Create a new Angular Project. Open the app.module.ts and import the ReactiveFormsModule

```

import { ReactiveFormsModule } from '@angular/forms';

```

Also, add it to the imports array of NgModule of the AppModule

```

@NgModule({
  imports: [BrowserModule, ReactiveFormsModule],
  declarations: [AppComponent, HelloComponent],
  bootstrap: [AppComponent]
})
export class AppModule {}

```

You can refer to the completed Source Code from StackBlitz

Import FormArray

Go to the AppComponent . Import the FormArray from the Angular Forms Module.

```

import { FormGroup, FormArray, FormBuilder } from '@angular/forms'

```

Build a Form Model

The First task is to build a Form Model `empForm` . It has only one property a FormArray of employees.

```

empForm:FormGroup;

constructor(private fb:FormBuilder) {
   this.empForm=this.fb.group({
     employees: this.fb.array([]) ,
   })
}

```

Employee FormArray

Helper method, which returns the `employees` FormArray from the model `empForm`

```

employees(): FormArray {
  return this.empForm.get("employees") as FormArray
}

```

The `newEmployee` method creates a new employee FormGroup and returns it. It has three properties. `firstName` , `lastName` and `skills` FormArray.

```

newEmployee(): FormGroup {
  return this.fb.group({
    firstName: '',
    lastName: '',
    skills:this.fb.array([])
  })
}

```

Next, the method to add an employee. It uses the `newEmployee` method which returns the `Employee` FormGroup and ads it to `employees` array.

```

addEmployee() {
  this.employees().push(this.newEmployee());
}

```

Method to remove the employee form the array. It needs the index position to remove it.

```

removeEmployee(empIndex:number) {
  this.employees().removeAt(empIndex);
}

```

Skills FormArray

Under each employee, we have skills array. Hence create helper method which returns a skills array from the employee array. We need to pass the index position of the employee array as argument.

```

employeeSkills(empIndex:number) : FormArray {
  return this.employees().at(empIndex).get("skills") as FormArray
}

```

`newSkill` method returns a `skill` FormGroup. It has two fields. Name of the `skill` and years of `exp`

```

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

```

```

addEmployeeSkill method the skill to employee.

```

addEmployeeSkill(empIndex:number) {
  this.employeeSkills(empIndex).push(this.newSkill());
}

```

Finally, removeEmployeeSkill method, which removes the skill of an employee.

```

removeEmployeeSkill(empIndex:number,skillIndex:number) {
  this.employeeSkills(empIndex).removeAt(skillIndex);
}

```

The onSubmit method accepts the Employee Forms. You can validate and update the database from here.

```

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

```

Template

Create a <form> and bind it to empForm using formgroup directive

```

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

```

Under empForm we have employees array. Bind it to the div element using formArrayName directive

```
<div formArrayName="employees">
```

Next, loop through the controls under the employees using ngFor. let empIndex=index will save the index position in the empIndex local variable.

```
  <div *ngFor="let employee of employees().controls; let empIndex=index">
```

The index is used as the name of the control in a Form Array. Hence use the [formGroupName]="empIndex" to bind it to the FormGroup . The style exists to show a nice border around employee

```
    <div [formGroupName]="empIndex" style="border: 1px solid blue; padding: 10px; widt
```

Input element for employee’s firstName , lastName . Also, place a button removeEmployee(empIndex) to remove this employee from the array.

```
    {{empIndex}}
      First Name :
      <input type="text" formControlName="firstName">
      Last Name:
      <input type="text" formControlName="lastName">

      <button (click)="removeEmployee(empIndex)">Remove</button>
```

Bind the skills of the empoyee to a div using formArrayName directive

```
<div formArrayName="skills">
```

ngFor, now loops through the `skills` array of the `employee`.

```
<div *ngFor="let skill of employeeSkills(empIndex).controls; let skillIndex=index">
```

input fields for skill and exp, Also button to remove the skill, which calls the `removeEmployeeSkill(empIndex,skillIndex)`

```

          <div [formGroupName]="skillIndex">
            {{skillIndex}}
            Skill :
            <input type="text" formControlName="skill">
            Exp:
            <input type="text" formControlName="exp">

            <button (click)="removeEmployeeSkill(empIndex,skillIndex)">Remove</button>

          </div>

```

Finally a button the add the skill to the employee

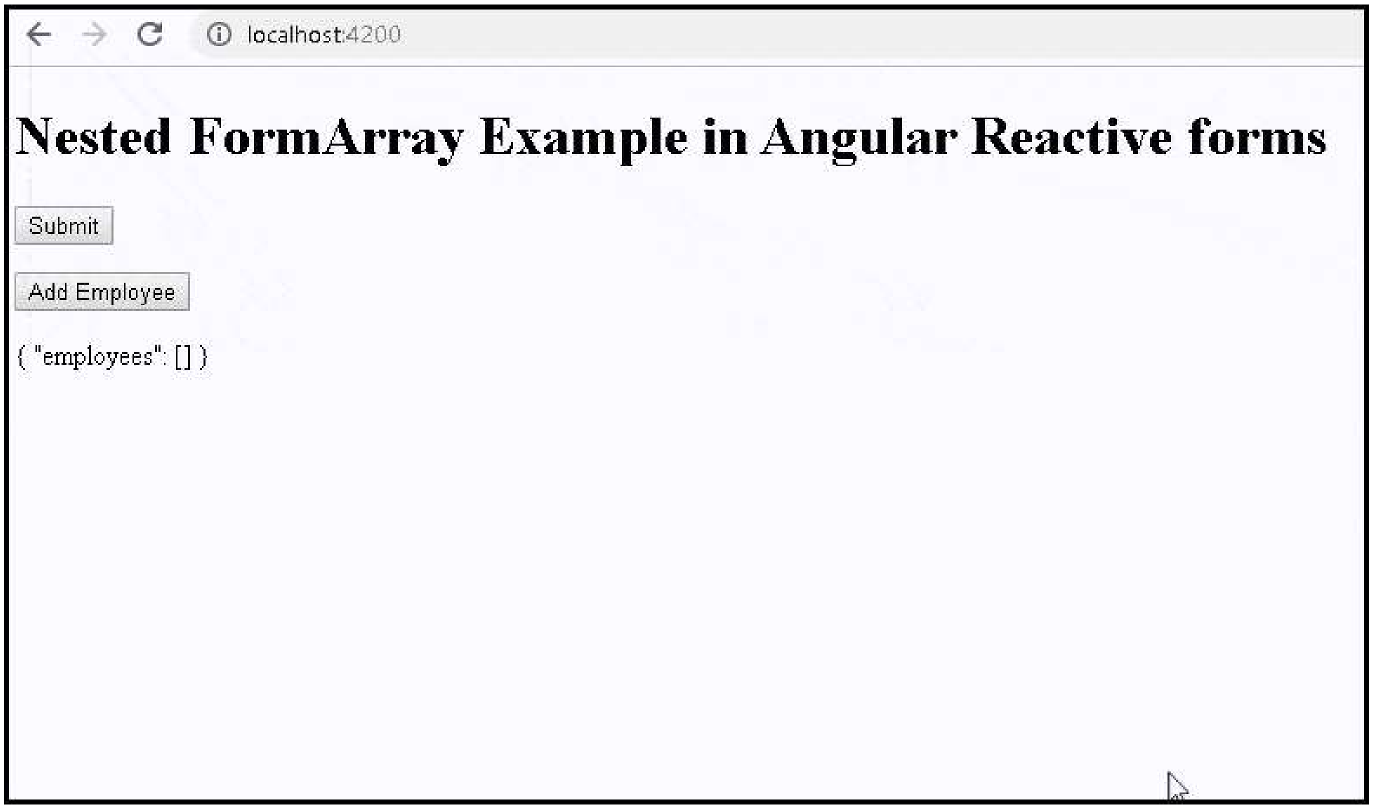
```
<button type="button" (click)="addEmployeeSkill(empIndex)">Add Skill</button>
```

Finally a button the add the employee

```

<button type="button" (click)="addEmployee()">Add Employee</button>

```



Nested FormArray Example. Add Form Fields Dynamically

Updating the Nested forms With Initial Data

Setting the initial values in the nested Forms is a little tricky. You need to build the form dynamically and then use the PatchValue & SetValue methods to update the form.

SetValue & PatchValue in FormArray Angular

Source Code

You can view the source code from the StackBlitz.

app.component.ts

```

import { Component, VERSION } from '@angular/core';
import { FormGroup, FormArray, FormBuilder } from '@angular/forms';

@Component({
  selector: 'my-app',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  empForm: FormGroup;

  constructor(private fb: FormBuilder) {}

  ngOnInit() {
    this.empForm = this.fb.group({
      employees: this.fb.array([])
    });
  }

  employees(): FormArray {
    return this.empForm.get('employees') as FormArray;
  }

  newEmployee(): FormGroup {
    return this.fb.group({
      firstName: '',
      lastName: '',
      skills: this.fb.array([])
    });
  }

  addEmployee() {
    this.employees().push(this.newEmployee());
  }

  removeEmployee(empIndex: number) {
    this.employees().removeAt(empIndex);
  }

  employeeSkills(empIndex: number): FormArray {
    return this.employees()
      .at(empIndex)
      .get('skills') as FormArray;
  }

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

  addEmployeeSkill(empIndex: number) {
    this.employeeSkills(empIndex).push(this.newSkill());
  }

  removeEmployeeSkill(empIndex: number, skillIndex: number) {
    this.employeeSkills(empIndex).removeAt(skillIndex);
  }

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


```

app.component.html

```

<h1>Angular Nested FormArray / Dynamic FormArray</h1>

<form [formGroup]="empForm" (ngSubmit)="onSubmit()">
  <div formArrayName="employees">
    <div *ngFor="let employee of employees().controls; let empIndex=index">
      <div
        [formGroupName]="empIndex"
        style="border: 1px solid blue; padding: 10px; width: 600px; margin: 5px;"
      >
        {{empIndex}} First Name :
        <input type="text" formControlName="firstName" />
        Last Name:
        <input type="text" formControlName="lastName" />

        <button (click)="removeEmployee(empIndex)">Remove</button>

        <div formArrayName="skills">
          <div
            *ngFor="let skill of employeeSkills(empIndex).controls; let skillIndex=index"
          >
            <div [formGroupName]="skillIndex">
              {{skillIndex}} Skill :
              <input type="text" formControlName="skill" />
              Exp:
              <input type="text" formControlName="exp" />

            <button (click)="removeEmployeeSkill(empIndex,skillIndex)">
              Remove
            </button>
          </div>
        </div>
      </div>
      <button type="button" (click)="addEmployeeSkill(empIndex)">
        Add Skill
      </button>
    </div>
  </div>
  <button type="button" (click)="addEmployee()">Add Employee</button>
 </div>
</form>

{{this.empForm.value | json}}

<br /><br />
<a
  href="https://www.tektutorialshub.com/angular/nested-formarray-example-add-form-fiel
  >Nested FormArray / Dynamic FormArray</a
>


```

Summary

In this tutorial, we learned how to create Multiple levels of nested forms in Angular.

Reference

- FormsArray

Read More

1. Angular Tutorial
2. Angular Forms Tutorial: Fundamental & Concepts
3. Template Driven Forms in Angular