ChatGPT - nqt

chatgpt.com/g/g-p-67b69ce152148191b0859bb5eba88ff8-nqt/c/67d83669-7ffc-8001-9f49-f3acd19875ef

Bài 5: Reactive Forms trong Angular

1. Giới thiệu về Reactive Forms

Reactive Forms là một cách tiếp cận mạnh mẽ và linh hoạt để xử lý biểu mẫu trong Angular. Không giống như Template-driven Forms, Reactive Forms hoạt động dựa trên lập trình hướng đối tượng và sử dụng API mạnh mẽ từ `FormControl`, `FormGroup`, và `FormArray`.

Lợi ích của Reactive Forms

✔️ Kiểm soát tốt hơn dữ liệu biểu mẫu trong TypeScript.
✔️ Xử lý và kiểm tra dữ liệu dễ dàng với `Validators`.
✔️ Có thể mở rộng với các nhóm và mảng dữ liệu linh hoạt.
✔️ Dễ dàng kiểm tra với unit test.

---

2. Cài đặt Reactive Forms trong Angular

Trước khi bắt đầu, bạn cần đảm bảo rằng `ReactiveFormsModule` đã được import vào `AppModule` hoặc module chứa component.

Cài đặt module

Mở file `app.module.ts` và thêm:

```
typescript
```

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import { ReactiveFormsModule } from '@angular/forms';

@NgModule({
  declarations: [AppComponent],
  imports: [
    BrowserModule,
    ReactiveFormsModule // Import ReactiveFormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

---

3. Tạo một Form Reactive Cơ Bản

Chúng ta sẽ tạo một form đơn giản gồm: **Tên**, **Email**, **Mật khẩu** với kiểm tra hợp lệ.

Bước 1: Khởi tạo form trong component

Trong `app.component.ts`, nhập các thư viện cần thiết và tạo form:

```
typescript
```

```
import { Component } from '@angular/core';
import { FormControl, FormGroup, Validators } from '@angular/forms';

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

  constructor() {
    this.userForm = new FormGroup({
      name: new FormControl('', [Validators.required,
Validators.minLength(3)]),
      email: new FormControl('', [Validators.required, Validators.email]),
      password: new FormControl('', [Validators.required,
Validators.minLength(6)])
    });
  }

  onSubmit() {
    if (this.userForm.valid) {
      console.log('Dữ liệu Form:', this.userForm.value);
    } else {
      console.log('Form không hợp lệ');
    }
  }
}
```

Bước 2: Thiết kế giao diện form

Mở `app.component.html` và thêm đoạn code sau:

```
html
```

```
<div class="container">
  <h2>Đăng ký tài khoản</h2>

  <form [formGroup]="userForm" (ngSubmit)="onSubmit()">
    <div>
      <label for="name">Tên:</label>
      <input id="name" type="text" formControlName="name">
      <span *ngIf="userForm.controls['name'].invalid &&
userForm.controls['name'].touched">
        Tên phải có ít nhất 3 ký tự
      </span>
    </div>

    <div>
      <label for="email">Email:</label>
      <input id="email" type="email" formControlName="email">
      <span *ngIf="userForm.controls['email'].invalid &&
userForm.controls['email'].touched">
        Email không hợp lệ
      </span>
    </div>

    <div>
      <label for="password">Mật khẩu:</label>
      <input id="password" type="password" formControlName="password">
      <span *ngIf="userForm.controls['password'].invalid &&
userForm.controls['password'].touched">
        Mật khẩu phải có ít nhất 6 ký tự
      </span>
    </div>

    <button type="submit" [disabled]="userForm.invalid">Đăng ký</button>
  </form>

  <p><strong>Dữ liệu form:</strong> {{ userForm.value | json }}</p>
</div>
```

Bước 3: Thêm CSS

Mở `app.component.css` và thêm:

```
css
```

```
.container {
  max-width: 400px;
  margin: auto;
  padding: 20px;
  border: 1px solid #ccc;
  border-radius: 5px;
  box-shadow: 0px 0px 10px #ddd;
}

input {
  display: block;
  width: 100%;
  padding: 8px;
  margin: 5px 0;
}

span {
  color: red;
  font-size: 12px;
}

button {
  padding: 10px;
  background-color: blue;
  color: white;
  border: none;
  cursor: pointer;
  margin-top: 10px;
}

button:disabled {
  background-color: gray;
}
```

4. Kiểm Tra Kết Quả

Chạy lệnh:

```
sh
```

```
ng serve
```

Mở trình duyệt và vào địa chỉ `http://localhost:4200/`, bạn sẽ thấy form đăng ký có kiểm tra dữ liệu.

---

5. Reactive Forms Nâng Cao

Bây giờ chúng ta sẽ mở rộng form với:

- **FormGroup lồng nhau** (Nested FormGroup).
- **FormArray** để thêm danh sách số điện thoại.

Bước 1: Cập nhật `app.component.ts`

Thay đổi form để có danh sách số điện thoại:

typescript

```typescript
import { Component } from '@angular/core';
import { FormArray, FormControl, FormGroup, Validators } from
'@angular/forms';

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

  constructor() {
    this.userForm = new FormGroup({
      name: new FormControl('', [Validators.required,
Validators.minLength(3)]),
      email: new FormControl('', [Validators.required, Validators.email]),
      password: new FormControl('', [Validators.required,
Validators.minLength(6)]),
      phones: new FormArray([])
    });
  }

  get phones(): FormArray {
    return this.userForm.get('phones') as FormArray;
  }

  addPhone() {
    this.phones.push(new FormControl('', Validators.required));
  }

  removePhone(index: number) {
    this.phones.removeAt(index);
  }

  onSubmit() {
    if (this.userForm.valid) {
      console.log('Dữ liệu Form:', this.userForm.value);
    } else {
      console.log('Form không hợp lệ');
    }
  }
}
```

Bước 2: Cập nhật `app.component.html`

Thêm phần danh sách số điện thoại:

```
html
```

```
<div>
  <label for="phones">Số điện thoại:</label>
  <div *ngFor="let phone of phones.controls; let i = index">
    <input type="text" [formControlName]="i">
    <button type="button" (click)="removePhone(i)">Xóa</button>
  </div>
  <button type="button" (click)="addPhone()">Thêm số điện thoại</button>
</div>
```

Tổng kết

- **Reactive Forms** giúp quản lý form một cách mạnh mẽ và linh hoạt.
- **Validators** hỗ trợ kiểm tra dữ liệu trực tiếp.
- **FormArray** giúp quản lý danh sách động như số điện thoại.
- **Reactive Forms phù hợp với ứng dụng phức tạp cần kiểm soát dữ liệu tốt.**

Bạn có thể tiếp tục mở rộng bài học bằng cách sử dụng **custom validators** hoặc **async validators**. Bạn có muốn tìm hiểu về chúng không? 🚀