academeez

@angular/forms

4. Reactive Forms

Reactive Forms

- With Reactive Forms I can describe any form structure using the classes: **FormControl, FormArray, FormGroup**
- After describing my form structure in the component class, I need to connect that structure to the template
- We connect the structure to the template using the directives:
  - **[formControl], [formControlName]**
  - **[formGroup], [formGroupName],**
  - **[formArray], [formArrayName]**

FormControl

- **FormControl** represents a single element used for grabbing user input.
- It can be attach to any html form element like: input, textarea, select, etc.
- We need to create the instance of the class in the component class
- We need to to attach the instance of the FormControl to our template using the directive **[formControl]**
- You can grab the value the user entered using the **value** property of the **FormControl**

FormControl - example

- 2 steps:
  1. Create instance of **FormControl**
  2. Attach the instance with the directive **[formControl]**

```
@Component({
  selector: 'name-input',
  template: `
    <input type="text" [formControl]="name" ↗
  `
})
export class NameInputComponent {
  name = new FormControl('initial value');
}
```

FormGroup

- In html we usually wrap the form elements in a form tag
- The form tag describes a group of form control elements
- A **FromGroup** is a class that also describes a group, you create an instance in the component class and attach it with a directive to the template
- You can nest sub groups
- You can grab the value of the entire group with the **value** property

FormGroup Example - Login Form

academeez

- Creating a login form which is a group that holds the email and password

```
@Component({
  selector: 'login',
  template: `
    <form [formGroup]="login">
      <input type="email" formControlName="email" ↗
      <input type="password" formControlName="password" ↗
    ↙form>

  `
})
export class LoginComponent {
  login = new FormGroup({
    email: new FormControl(),
    password: new FormControl()
  })
}
```

FormGroup Example - Sub group

- A group can contain sub group which can contain sub group and so on.
- For example a register form might want to create a group for the password and repeat password fields

```
@Component({
  selector: 'register',
  template: `
    <form [formGroup]="register">
      <input type="email" formControlName="email" ↗
      <div formGroupName="passwordRepeat">
        <input type="password" formControlName="password" ↗
        <input type="password" formControlName="repeat" ↗
      ↙div>
    ↙form>
  `
})
```

```
export class RegisterComponent {
  register = new FormGroup({
    email: new FormControl(),
    passwordRepeat: new FormGroup({
      password: new FormControl(),
      repeat: new FormControl()
    })
  })
}
```

FormGroup dynamic - EX

- With Reactive Forms, you forms can be dynamic
- For example you can place a condition on your form group, if a certain FormControl or sub FormGroup is presented than display them on the form
- Let’s try to create a simple for with a button that will toggle a text input

FormArray

- Our forms can contain repeating elements.
- The repeating elements can be groups, or single controls, or nested arrays.
- To represet something that repeats in our form we use the **FormArray**
- The **FormArray** elements are often dynamic and you can add more elements to the **FormArray**
- You can connect the **FormArray** with the following directives: **[formArray], [formArrayName]**
- You will also need to connect the elements in the array

FormArray - example

- The following form contain an array with multiple FormControl

```
@Component({
  selector: 'form-array',
  template: `
    <form [formGroup]="example">
      <div formArrayName="sampleArray">
        <input
          type="text"
          *ngFor="let arrayInput of example.controls.sampleArray.controls"
          [formControl]="arrayInput"
        ↗
      ↙div>
    ↙form>
  `
})
```

```
export class SampleComponent {
  example = new FormGroup({
    sampleArray: new FormArray([
      new FormControl()
    ])
  })
}
```

Dynamic FormArray - ex

- Create a form where the user can enter address containing country and city
- The address is a group
- A user can have multiple addresses (array)
- There is a button in the form for adding another address

FormBuilder

academeez

- The **FormBuilder** service is a shortcut for creating our **FormGroup, FormArray, FormControl**
- With the service you don’t need to create instances of **FormGroup, FormArray, FormControl** instead you call the methods:
  - formBuilder.control
  - formBuilder.array
  - formBuilder.group
- On simple form you can place a class instance of your model inside **formBuilder.group** and it will create the proper form elements you can attach to the form in the template

FormBuilder - ex

- In the following example we are building a form group with nested sub group using **FormBuilder**

```
export class BuilderExampleComponent {
  register = this._formBuilder.group({
    email: '',
    passwordRepeat: this._formBuilder.group({
      password: '',
      repeat: ''
    })
  })

  constructor(private _formBuilder: FormBuilder) {}
}
```

Summary

- With the Reactive Forms we are building our form in the component class using, FormControl, FormGroup, FormArray
- We then attach the form to the template using the directives: [formControl], [formGroup], [formArray], [formControlName], [formGroupName], [formArrayName]
- The form can be dynamic, if we change the structure of the form in the component class we can reflect that dynamic change in the template
- We can utilise FormBuilder to construct our form code easier.

academeez

Thank You

Next Lesson: 5. Validation