Tipo : Guía del Enunciado
Capitulo : Creando Aplicaciones ASP.NET Core
Duración : 180 minutos

I. OBJETIVO

Model Driven Forms

II. REQUISITOS

Los siguientes elementos de software son necesarios para la realización del laboratorio:

1. Windows 10 (como mínimo Windows 8)
2. Visual Studio Code

III. EJECUCIÓN DEL LABORATORIO

1. Si queremos implementar validaciones personalizadas, necesitamos utilizar “model driven form”.
2. Creamos un componente llamado “login”, y agregamos en el sidebar la opcion.
3. poner el siguiente codigo en el html del componente:

```
<form>
 <div class="form-group">
   <label for="username">UserName</label>
   <input id="username" type="text" class="form-control">
 </div>
 <div class="form-group">
   <label for="password">Password</label>
   <input id="password" type="password" class="form-control">
 </div>
 <button class="btn btn-primary" type="submit">Login</button>
</form>
```

El codigo de arriba es solo html5 con unas pocas clases de bootstrap, tampoco tiene ninguna directiva de angular, actualizaremos nuestro formulario para implementar validaciones personalizadas en el username y password.

4. En el componente agregar el codigo resaltado:

```
import { Component, OnInit } from '@angular/core';
import { FormGroup, FormControl,Validators } from '@angular/forms';
@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent  {

  form = new FormGroup({
    username: new FormControl('', Validators.required),
    password: new FormControl('', Validators.required)
  });

}
```

5. En el modulo de cliente, debe estar importado estos dos modulos:

```
import { NgModule } from '@angular/core';
import { SharedModule } from './../shared/shared.module';
import { ClienteListaComponent } from './cliente-lista/cliente-lista.component';
import { ClienteRoutingModule } from './cliente-routing.module';
import { MatTableModule, MatListModule, MatToolbarModule, MatButtonModule, MatProgressS
import { ClienteListaService } from './cliente-lista/cliente-lista.service';
import { CommonModule } from '@angular/common';
import { SidebarComponent } from './sidebar/sidebar.component';
import { ClienteContainerComponent } from './cliente-container/cliente-container.compor
import { ClienteTableComponent } from './cliente-table/cliente-table.component';
import { LoginComponent } from './login/login.component';
import {ReactiveFormsModule,FormsModule} from '@angular/forms';
@NgModule({
    imports: [
        ReactiveFormsModule,
        FormsModule,
        CommonModule,
```

6. Regresamos al template, necesitamos decirle a Angular que creamos un “FormGroup” y su “FormControls”.

```
<form [formGroup]="form">
  <div class="form-group">
    <label for="username">UserName</label>
    <input id="username" type="text" class="form-control" formControlName="username">
  </div>
  <div class="form-group">
    <label for="password">Password</label>
    <input id="password" type="password" class="form-control" formControlName="password">
  </div>
  <button class="btn btn-primary" type="submit">Login</button>
</form>
```

En el elemento “form”, aplicamos la directiva “formgroup” para bindearlo a “form”. ahora definiremos las validaciones:

```
<form [formGroup]="form">
  <div class="form-group">
    <label for="username">UserName</label>
    <input id="username" type="text" class="form-control" formControlName="username">
    <div *ngIf="form.controls.username.touched && !form.controls.username.valid"
    class="alert alert-danger">
      UserName is Required.
    </div>
  </div>
  <div class="form-group">
    <label for="password">Password</label>
    <input id="password" type="password" class="form-control" formControlName="password">
    <div *ngIf="form.controls.password.touched && !form.controls.password.valid"
     class="alert alert-danger">
      Password is Required.
    </div>
  </div>
  <button class="btn btn-primary" type="submit">Login</button>
</form>
```

7. Para manejar el evento “submit” del form. Bindear el evento ngSubmit a el metodo login() en el form.

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

8. Ahora, implementaremos el metodo “login()”

```
login(){
  console.log(this.form.value)// imprimir los valores del form en formato json
}
```

9. Ejecutar y probar, lo que hemos hecho hasta el momento es similar a template driven forms. la diferencia es que en el model driven forms, explicitamente creamos el “formGroup” y “FormControl”, en Template driven forms, angular lo crea por nosotros.

10. Usaremos FormBuilder:

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

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent {

  // form = new FormGroup({
  //   username: new FormControl('', Validators.required),
  //   password: new FormControl('', Validators.required)
  // });
  form: FormGroup;
  constructor(fb: FormBuilder) {
    this.form = fb.group({
      username: ['', Validators.required],
      password: ['', Validators.required]
    });
  }

  login() {
    console.log(this.form.value)// imprimir los valores del form en formato json
  }
}
```

11. Agregamos una nueva carpeta en el modulo de cliente y agregamos un archivo llamado “passwordValidator.ts”

ogin.component.css | login.component.html | login.component.ts | passwordValidator.ts

```
import { FormControl } from '@angular/forms';

export class PasswordValidator {
    static cannotContainSpace(formControl: FormControl) {
        if (formControl.value.indexOf(' ') >= 0)
            return { cannotContainSpace: true };

        return null;
    }
}
```

12. en login component, hacemos la siguiente validacion:

```ts
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { PasswordValidator } from './../validators/passwordValidator';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent {

  // form = new FormGroup({
  //   username: new FormControl('', Validators.required),
  //   password: new FormControl('', Validators.required)
  // });
  form: FormGroup;
  constructor(fb: FormBuilder) {
    this.form = fb.group({
      username: ['', Validators.required],
      password: ['', Validators.compose([Validators.required, PasswordValidator.cannotContainSpace])]
    });
  }

  login() {
    console.log(this.form.value)// imprimir los valores del form en formato json
  }
}
```

13. modificamos el html:

ogin.component.css | login.component.html | login.component.ts | passwordValidator.ts

```html
<form [formGroup]="form" (ngSubmit)="login()">
  <div class="form-group">
    <label for="username">UserName</label>
    <input id="username" type="text" class="form-control" formControlName="username">
    <div *ngIf="form.controls.username.touched && !form.controls.username.valid" class="alert alert-danger">
      UserName is Required.
    </div>
  </div>
  <div class="form-group">
    <label for="password">Password</label>
    <input id="password" type="password" class="form-control" formControlName="password">
    <div *ngIf="form.controls.password.touched && form.controls.password.errors">
      <div *ngIf="form.controls.password.errors.required" class="alert alert-danger">
        Password is required.
      </div>
      <div *ngIf="form.controls.password.errors.cannotContainSpace" class="alert alert-danger">
        Password cannot contain space
      </div>
    </div>
  </div>
  <button class="btn btn-primary" type="submit">Login</button>
</form>
```

14. Creamos un servicio:

```
ng g service cliente/login/login
```

```
import { Injectable } from '@angular/core';

@Injectable()
export class LoginService {

  constructor() { }
  login(username, password) {
    if (username === "cibertec" && password === "123") {
      return true
    }else{
      return false;
    }
  }
}
```

15. Actualizamos el login component.

```
  styleUrls: ['./login.component.css']
})
export class LoginComponent {

  // form = new FormGroup({
  //   username: new FormControl('', Validators.required),
  //   password: new FormControl('', Validators.required)
  // });
  form: FormGroup;
  constructor(fb: FormBuilder, private _loginService: LoginService) {
    this.form = fb.group({
      username: ['', Validators.required],
      password: ['', Validators.compose([Validators.required, PasswordValidator.cannotContainSpace])]
    });
  }

  login() {
    var result =
      this._loginService.login(this.form.controls['username'].value, this.form.controls['password'].value);

    if (!result) {
      this.form.controls['password'].setErrors({ invalidLogin: true })
    }
  }
}
```

16. Finalmente actualizamos el html

```
<form [formGroup]="form" (ngSubmit)="login()">
  <div class="form-group">
    <label for="username">UserName</label>
    <input id="username" type="text" class="form-control" formControlName="username">
    <div *ngIf="form.controls.username.touched && !form.controls.username.valid" class="alert alert-danger">
      UserName is Required.
    </div>
  </div>
  <div class="form-group">
    <label for="password">Password</label>
    <input id="password" type="password" class="form-control" formControlName="password">
    <div *ngIf="form.controls.password.touched && form.controls.password.errors">
      <div *ngIf="form.controls.password.errors.invalidLogin" class="alert alert-danger">
        UserName or password is invalid.
      </div>
      <div *ngIf="form.controls.password.errors.required" class="alert alert-danger">
        Password is required.
      </div>
      <div *ngIf="form.controls.password.errors.cannotContainSpace" class="alert alert-danger">
        Password cannot contain space
      </div>
    </div>
  </div>
  <button class="btn btn-primary" type="submit">Login</button>
</form>
```