Harmony IT Solution

Angular

Tahaluf Training Center 2022

Harmony IT Solution

1. Routing.
2. Default Route.
3. Create and Validate Form.

Harmony IT Solution

Objective

The Objective of this lecture

- Understand the routing in the angular and how to navigate from one component to another.
- Get to know about the reactive form.
- How to deal with the form control and apply the validation for each input.

Routing

Overview of Routing

Routing allows you to move from one part of the application to another part or one View to another View.

Overview of Routing

Since the app component is the first component that is loaded when the project
Has run and if you want to go to another page we will use routing in the app component.

Add this tag in the app.component.html:

```
<router-outlet></router-outlet>
```

Example of Routing

Generate a new component called about us and another component called contact us in the app module.

Example Solution

Generate about us component in the app module:

```
PS C:\Users\d.kanaan.ext\Desktop\EduTech> ng g c aboutus
CREATE src/app/aboutus/aboutus.component.html (22 bytes)
CREATE src/app/aboutus/aboutus.component.spec.ts (633 bytes)
CREATE src/app/aboutus/aboutus.component.ts (279 bytes)
CREATE src/app/aboutus/aboutus.component.css (0 bytes)
UPDATE src/app/app.module.ts (739 bytes)
```

Overview of Routing

The first page will load is app components and if you want to go to the about us component in the same module **(App Module).**

In-app**-routing.module.ts,** There is an array called **routes**
this array is used to add the route for all components.

Overview of Routing

Each route is an object and each route consists of a path
and component

path: where you want to go for example /aboutus but you must write the page name without using /.

component: component name which you want to display.

Example of Routing

```
const routes: Routes = [
  {
      path: 'aboutus',
      component: AboutusComponent
  }
];
```

NOTE: Once you write the component name it will import this component like this:

```
import { AboutusComponent } from './aboutus/aboutus.component';
```

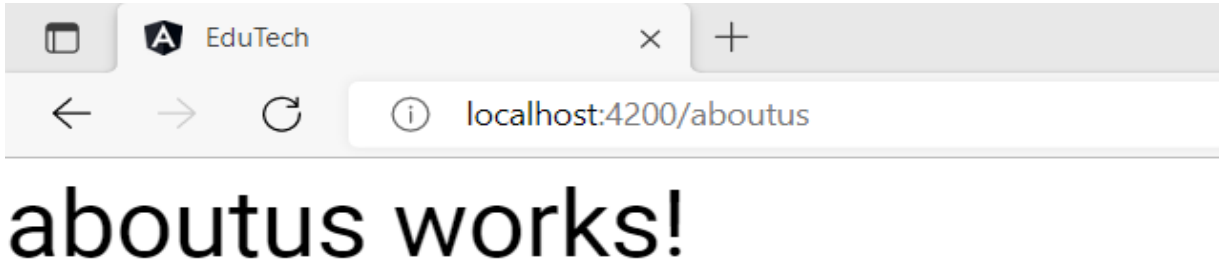
Example of Routing

To add more than one route separate them with a comma.

```
const routes: Routes = [
  {
      path: 'aboutus',
      component: AboutusComponent
  },
  {
      path: 'contactus',
      component: ContactusComponent
  }
];
```

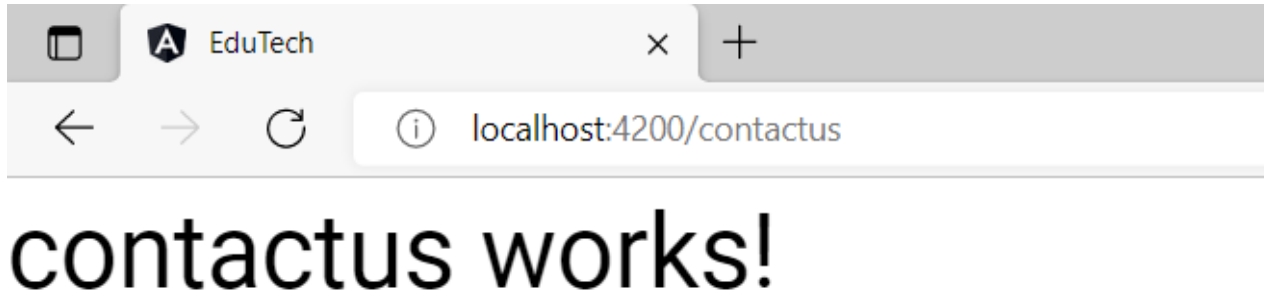
Example of Routing

Because about us component in the app module use /aboutus in the URL.



Example of Routing

Contact us component.



Default Route

Overview of Routing

The default route is redirected -for example- to the home path.

This means that, when you navigate to the root of your application /, you are redirected to the home path (/home).

The path is empty, indicating the default route.

Example of Routing

The default route of the project is the about us component.

```
const routes: Routes = [
  {
      path: '',
      component: AboutusComponent
  },
]
```

In this way when the project was run the about us component will load.

Exercise

Create a route for auth module to login and register components.

Example of Routing

If you want to make the login page the first page to be loaded.

In-app-routing.module.ts

```
const routes: Routes = [
  {
      path: '',
      loadChildren: () => import('./auth/auth.module').
          then((m) => m.AuthModule)
  },]
```

Example of Routing

And In auth-routing.module.ts :

```
const routes: Routes = [
  {
      path: '',
      component: LoginComponent
  },
  {
      path: 'register',
      component: RegisterComponent
  }
];
```

Reactive Forms

Overview of Reactive Form

Many common applications handle input from users with forms.

Applications use forms to enable users to log in, update a profile, enter sensitive information, and perform many other tasks.

Overview of Reactive Form

Angular supports two ways of handling user input through forms: reactive and template-driven.

It captures user input events from the view, validates the input, creates a form model and data model, and tracks changes.

Overview of Reactive Form

To generate an internal representation of a template-driven form, template directives are used.

In reactive forms, you create your own form representation in the component class.

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

imports: [
    CommonModule,
    AuthRoutingModule,
    ReactiveFormsModule
]
```

Advantages of Reactive Form

1. Using custom validators.
2. Changing validation dynamically.
3. Dynamically adding form fields.

Harmony IT Solution

Form Control

Form Group

Harmony IT Solution

Form Control

Overview of Form Control

Form Control: Form input instance with multiple functionalities, ex: validation, flag, and data binding.

An Angular form control encapsulates both the value of data and the validation information for each form element.

In a reactive form, every input should be bound to a form control.

Syntax of Form Control

Use this syntax to create an instance of the form control class:

```
Instance_name = new FormControl(form state, Validation Option)
```

Form Group

Overview of Form Group

A form group encapsulates a collection of form controls.

Controls give you access to the state of elements, while groups give you access to the state of wrapped controls.

At initialization, every form control in a form group is identified by its name.

How to deal with a reactive form

To work with reactive forms, you will be using the ReactiveFormsModule instead of the FormsModule.

Step 1:
Open the auth.module.ts and add ReactiveFormsModule.

How to deal with a reactive form

Step 2:
Adding a Form to the Component Template.

Step 3:
Building the Component Class.

How to deal with a reactive form

We will use angular material, To use angular material in your project you must install this package:

ng add @angular/material

Angular Material Website:
https://material.angular.io/

Overview of Angular Material

A Library that developers can use in their Angular projects to speed up the development of consistent user interfaces.

Angular Material offers you reusable and beautiful UI components like Cards, Inputs, Forms, and much more.

Harmony IT Solution

Create & Validate Form

How to create and validate a form in angular

We will start working on LoginComponent and create the login form using the form control.

Login form template link:
https://codepen.io/paulhbarker/pen/mwOREz

How to create and validate a form in angular

To use form filed from angular material, you must add MatFormFieldModule in the import array in auth.module.ts

```
import {MatFormFieldModule} from  '@angular/material/form-field’;

imports: [
    CommonModule,
    AuthRoutingModule,
    ReactiveFormsModule,
    MatFormFieldModule
]
```

How to create and validate a form in angular

To use the input component from angular material in auth.module.ts, add MatInputModule in the import array.

```
import {MatInputModule} from  '@angular/material/input’;

imports: [
    CommonModule,
    AuthRoutingModule,
    ReactiveFormsModule,
    MatFormFieldModule
]
```

Example of the Form Control

In login.component.ts

```
export class LoginComponent implements OnInit {
  constructor(private router:Router,public authServices:AuthService ) { }
  email = new FormControl('',[Validators.required,Validators.email]);
  password=new FormControl('',[Validators.required,Validators.minLength(8)]);
  ngOnInit(): void {
  }
  submit(){
    console.log(this.email.value);
    console.log(this.password.value)
  }

  goToRegister()
  {
    this.router.navigate(['security/register'])
  }
}
```

Example of the Form Control

In login.component.html

```
<div id="login">
  <div class="login-card">
    <div class="card-title">
      <h1>Please Sign In</h1>
    </div>
    <div class="content" style="width:50;">
      <form method="POST" action="#">
        <mat-form-field class="example-full-width">
        <input matInput id="email" type="email"
        [formControl]="email" title="email" placeholder="Email" >
        <mat-error *ngIf="email.hasError('email')">
          Please enter a valid email address
        </mat-error>
        <mat-error *ngIf="email.hasError('required')">
          Email is <strong>required</strong>
        </mat-error>
      </mat-form-field>
```

Example of the Form Control

In login.component.html

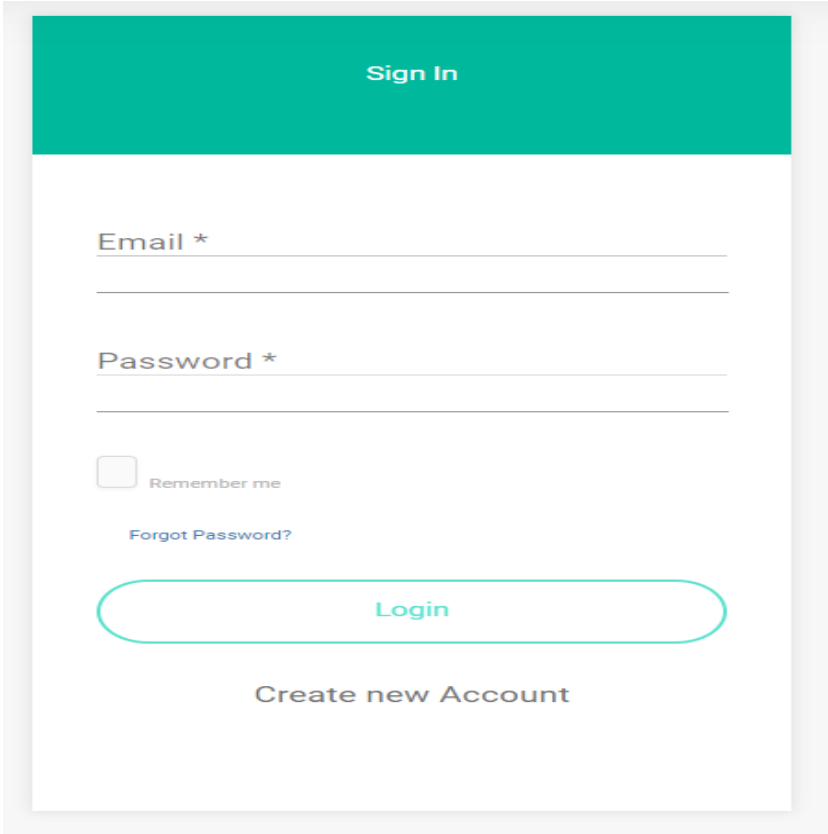
```
<mat-form-field class="example-full-width">
  <input matInput id="password"  [formControl]="password"
  type="password" title="password" placeholder="Password" >
  <mat-error *ngIf="password.hasError('minlength')">
    Please enter a strong password
  </mat-error>
  <mat-error *ngIf="password.hasError('required')">
      password is <strong>required</strong>
  </mat-error>
</mat-form-field>
  <div class="level options">
    <div class="checkbox level-left">
      <input type="checkbox" id="checkbox" class="regular-checkbox">
      <label for="checkbox"></label>
      <span>Remember me</span>
    </div>

    <a class="btn btn-link level-right" href="#">Forgot Password?</a>
  </div>
```

Example of the Form Control

In login.component.html

```
            <button type="submit" [disabled]="!email.valid || !password.valid"
              (click)="submit()" class="btn btn-primary">Login</button>
          <br>
            <p style="cursor:pointer; text-align: center;"
            (click)="goToRegister()">
            Create new Account </p>
          </form>
        </div>
      </div>
    </div>
```

The Output:



How does the ngIf work?

In login.component .ts

```
showMessage = false;

  toggleShowMessage() {
    this.showMessage = !this.showMessage;
  }
```

How does the ngIf work?

In login.component .html

```
<h4 *ngIf="showMessage">This is message to show</h4>
<!-- ifshowMessage==true then 'This is message to show' else '' -->
<button (click)="toggleShowMessage()">
    toggle</button>
```

References

[1] Angular, “Angular,” *Angular.io*, 2019. https://angular.io/

[2] “Complete Angular Tutorial For Beginners,” *TekTutorialsHub*. https://www.tektutorialshub.com/angular-tutorial/

[3]“npm | build amazing things,” *Npmjs.com*, 2019. https://www.npmjs.com/

[4]“Angular Tutorial for Beginners | Simplilearn,” *Simplilearn.com*. https://www.simplilearn.com/tutorials/angular-tutorial (accessed Aug. 19, 2022).

Harmony IT Solution