deti universidade de aveiro
departamento de electrónica,
telecomunicações e informática

Technologies and Web Programming

Angular Framework

Angular Framework

Services and Dependency Injection

Services

- A component should not need to define things like:
  - how to fetch data from the server;
  - validate user input;
  - or log directly to the console.

- Ideally, its job is to enable the user experience and nothing more. It must present properties and methods for data binding, in order to mediate between the view and the application logic.
- Instead, it should delegate such tasks to other type of processors, named Services in Angular.

Services

- A Service is a broad category encompassing any value, function, or feature that an app needs.
- A service is typically a class with a narrow, well-defined purpose. It should do something specific and do it well.
- By defining that kind of processing task in an injectable service class, you make it available to any component.
- Apps can also be more adaptable by injecting different providers of the same kind of service, as appropriate in different circumstances.

Dependency Injection – DI

- Dependency injection (often called DI) is wired into the Angular Framework and used everywhere to provide new components with the services or other things they need.
- Components consume services; that is, a service can be injected into a component, giving the component access to that service class.
- To define a class as a service in Angular, it uses the *@Injectable* decorator to provide the metadata that allows Angular to inject it into a component as a dependency.

Creating a Service (i)

- Run the following command-line:
  - ng generate service author

author.service.ts

```
1   import { Injectable } from '@angular/core';
2
3   @Injectable({  Show usages
4     providedIn: 'root'
5   })
6   export class AuthorService {
7     private baseUrl : string  = 'http://localhost:8000/ws/';
8
9     constructor() { }  no usages
10  }
```

Creating a Service (iv)

- Getting authors.

```
12    async getAuthor(id: number): Promise<Author> {  Show usages
13      const url = `${this.baseUrl}/author?id=${id}`;
14      const data : Response  = await fetch(url);
15      return await data.json() ?? undefined;
16    }
17
18    async getAuthors(): Promise<Author[]> {  Show usages
19      const url = `${this.baseUrl}/authors`;
20      const data : Response  = await fetch(url);
21      return await data.json() ?? [];
22    }
23
24    async getAuthorsN(num: number): Promise<Author[]> {  Show usages
25      const url = `${this.baseUrl}/authors?num=${num}`;
26      const data : Response  = await fetch(url);
27      return await data.json() ?? [];
28    }
```

Creating a Service (v)

- Changing authors.

```
30    async createAuthor(au: Author): Promise<any> {  no usages
31      const url = `${this.baseUrl}/authorcre`;
32      const data : Response  = await fetch(url, {
33        method: "POST", headers: new Headers({"Content-Type": "application/json"}), body: JSON.stringify(au)});
34      return await data.json();
35    }
36
37    async updateAuthor(au: Author): Promise<any> {  Show usages
38      const url = `${this.baseUrl}/authorupd`;
39      const data : Response  = await fetch(url, {
40        method: "PUT", headers: new Headers({"Content-Type": "application/json"}), body: JSON.stringify(au)});
41      return await data.json();
42    }
43
44    async deleteAuthor(au: Author): Promise<any> {  Show usages
45      const url = `${this.baseUrl}/authordel/${au.id}`;
46      const data : Response  = await fetch(url, {
47        method: "DELETE", headers: new Headers({"Content-Type": "application/json"}), body: JSON.stringify(au)})
48      return await data.text();
49    }
```

Components

- Changes in “top.ts”

top.ts

```ts
import { Component, inject } from '@angular/core';
import { RouterLink } from '@angular/router';
import { AsyncPipe } from '@angular/common';
import { Author } from '../author';
import { AuthorService } from '../author-service';


@Component({  Show usages
  selector: 'app-top',
  imports: [RouterLink, AsyncPipe],
  templateUrl: './top.html',
  styleUrl: './top.css',
})
export class Top {
  authorService: AuthorService = inject(AuthorService);
  authors$: Promise<Author[]> = this.authorService.getAuthorsN( num: 4);
}
```

Components

- Changes in “top.html”

<> top.html ×

```
1   <h2>Top Authors</h2>
2
3   <div class="grid grid-pad">
4     @for (au of authors$ | async; track au.id) {
5       <a class="col-1-4">
6           <div class="module author" routerLink="/authordetails/{{au.id}}">
7             <h4>{{au.name}}</h4>
8           </div>
9       </a>
10    }
11  </div>
```

Components

- Changes in “authors.ts”

authors.ts

```
1   import { Component, inject } from '@angular/core';
2   import { AsyncPipe } from '@angular/common';
3   import { FormsModule } from '@angular/forms';
4   import { RouterLink } from '@angular/router';
5   import { Author } from '../author';
6   import { AuthorService } from '../author-service';
7
8
9
10  @Component({  Show usages
11    selector: 'app-authors',
12    imports: [FormsModule, RouterLink, AsyncPipe],
13    templateUrl: './authors.html',
14    styleUrl: './authors.css',
15  })
16  export class Authors {
17    authorService: AuthorService = inject(AuthorService);
18    authors$: Promise<Author[]> = this.authorService.getAuthors();
19  }
```

Components

- Changes in “authors.html”

authors.html

```
1   <h2>Authors</h2>
2   <ul class="authors">
3     @for (au of authors$ | async; track au.id) {
4       <li>
5         <a routerLink="/authordetails/{{au.id}}">
6           <span class="badge">{{ au.id }}</span> {{au.name}}
7         </a>
8       </li>
9     }
10  </ul>
```

Components

- Changes in “author-details.html”

<> author-details.html ×

```
1  @if (author$ | async; as author : Author | null ) {
2    <h2>Information on {{author.name | uppercase}}</h2>
3    <div>Id: {{author.id}}</div>
4    <div>
5      <label>
6        Id:
7        <input [(ngModel)]="author.id" readonly />
8      </label>
9    </div>
10   <div>
11     <label>
12       Name:
13       <input [(ngModel)]="author.name" placeholder="name" />
14     </label>
15   </div>
16   <div>
17     <label>
18       Email:
19       <input [(ngModel)]="author.email" placeholder="email" />
20     </label>
21   </div>
22   <div>
23     <button (click)="update()">Update</button><br />
24     <button (click)="delete()">Delete</button><br />
25     <button (click)="goBack()">Go Back</button>
26   </div>
27 }
```

Components

- Changes in “author-details.ts”

author-details.ts

```
1   import { Component, inject } from '@angular/core';
2   import { FormsModule } from "@angular/forms";
3   import { AsyncPipe } from '@angular/common';
4   import { UpperCasePipe, Location } from "@angular/common";
5   import { ActivatedRoute } from '@angular/router';
6   import { Author } from '../author';
7   import { AuthorService } from '../author-service';
8
9   @Component({  Show usages
10    selector: 'app-author-details',
11    imports: [FormsModule, UpperCasePipe, AsyncPipe],
12    templateUrl: './author-details.html',
13    styleUrl: './author-details.css',
14  })
15  export class AuthorDetails {
16    authorService: AuthorService = inject(AuthorService);
17    author$: Promise<Author> | null = null;
18
19    constructor(private route: ActivatedRoute, private location: Location,) {
20      this.author$ = this.getAuthor();
21    }
```

Components

- Changes in “author-details.ts”

author-details.ts

```
15  export class AuthorDetails {
23    getAuthor(): Promise<Author> | null {  Show usages
24      let id : any  = this.route.snapshot.params['id'];
25      if (id == null) return null;
26      id = +id;
27      return this.authorService.getAuthor(id);
28    }
29
30    async update(): Promise<void> {  Show usages
31      if (this.author$ != null) {
32        const author : Author  = await this.author$;
33        const data : any  = await this.authorService.updateAuthor(author);
34        console.log(data);
35        this.goBack();
36      }
37    }
38
39    async delete(): Promise<void> {  Show usages
40      if (this.author$ != null) {
41        const author : Author  = await this.author$;
42        const data : any  = await this.authorService.deleteAuthor(author);
43        console.log(data);
44        this.goBack();
45      }
46    }
47
48    goBack(): void {  Show usages
49      this.location.back();
50    }
51  }
```