deti universidade de aveiro
departamento de electrónica,
telecomunicações e informática

Tecnologias e Programação Web
2019/2020

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.

Configuration

- Add the following lines to “app.module.ts” file:

app.module.ts

```
13  import {HttpClientModule} from '@angular/common/http';
14
15  @NgModule({
16    declarations: [
17      AppComponent,
18      AuthorsComponent,
19      OverviewComponent,
20      AuthorDetailsComponent
21    ],
22    imports: [
23      BrowserModule,
24      FormsModule,
25      HttpClientModule,
26      AppRoutingModule
27    ],
28    providers: [
29    ],
30    bootstrap: [AppComponent]
31  })
32  export class AppModule { }
```

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({
4    providedIn: 'root',
5  })
6  export class AuthorService {
7
8    constructor() { }
9
10 }
```

Creating a Service (ii)

- Modifications to access DB through REST API.

author.service.ts

```
1
2   import { Injectable } from '@angular/core';
3   import { Author } from './author';
4   import { Observable } from 'rxjs/internal/Observable';
5   import { HttpClient, HttpHeaders } from '@angular/common/http';
6
7   const httpOptions = {
8     headers: new HttpHeaders( headers: { 'Content-Type': 'application/json' })
9   };
10
11  @Injectable({
12    providedIn: 'root',
13  })
14  export class AuthorService {
15    private baseUrl = 'http://localhost:8000/ws/';
16
17    constructor(private http: HttpClient) { }
```

Observables

- Angular's HttpClient methods return RxJS Observables.
- Observables provide support for passing messages between publishers and subscribers in your application.
- They offer significant benefits over other techniques for event handling, asynchronous programming, and handling multiple values.
- Observables are declarative - that is, you define a function for publishing values, but it is not executed until a consumer subscribes to it. The subscribed consumer then receives notifications until the function completes, or until they unsubscribe.

Creating a Service (iii)

- Getting authors.

author.service.ts

```
19
20    getAuthorId(id: number): Observable<Author>  {
21      const url = this.baseUrl + 'author?id=' + id;
22      return this.http.get<Author>(url);
23    }
24
25
26    getAuthors(): Observable<Author[]>  {
27      const url = this.baseUrl + 'authors';
28      return this.http.get<Author[]>(url);
29    }
30
31    getAuthorsN(num: number): Observable<Author[]>  {
32      const url = this.baseUrl + 'authors?num=' + num;
33      return this.http.get<Author[]>(url);
34    }
```

Creating a Service (iv)

- Changing authors.

author.service.ts

```
37    createAuthor(au: Author): Observable<Author> {
38      const url = this.baseUrl + 'authorcre';
39      return this.http.post(url, au, httpOptions);
40    }
41
42
43    updateAuthor(au: Author): Observable<any> {
44      const url = this.baseUrl + 'authorupd';
45      return this.http.put(url, au, httpOptions);
46    }
47
48
49    deleteAuthor(au: Author): Observable<any> {
50      const url = this.baseUrl + 'authorch/' + au.id;
51      return this.http.delete<Author>(url, httpOptions);
52    }
```

Components

- Changes in “overview.component.html”

overview.component.html

```
1   <h3>Top Authors</h3>
2   <p></p>
3   <!-- <div class="grid grid-pad"> -->
4   <div class="row">
5     <a *ngFor="let author of authors" class="col" routerLink="/details/{{author.id}}">
6       <!-- <div class="module author"> -->
7       <div class="btn btn-primary btn-block">
8         <h4>{{author.name}}</h4>
9       </div>
10    </a>
11  </div>
```

Components

- Changes in “overview.component.ts”

overview.component.ts

```
@Component({
  selector: 'app-overview',
  templateUrl: './overview.component.html',
  styleUrls: ['./overview.component.css']
})
export class OverviewComponent implements OnInit {
  authors: Author[];

  constructor(private authorService: AuthorService) {
  }

  ngOnInit() {
    this.getAuthors();
  }

  getAuthors(): void {
    this.authorService.getAuthorsN( num: 4).subscribe( next: authors => this.authors = authors);
  }
}
```

Components

- Changes in “authors.component.html”

authors.component.html

```
1
2   <h2>Authors</h2>
3   <ul class="authors">
4     <li *ngFor="let author of authors">
5       <a routerLink="/details/{{author.id}}">
6         <span class="badge">{{ author.id }}</span> {{ author.name }}
7       </a>
8     </li>
9   </ul>
```

Components

- Changes in “authors.component.ts”

authors.component.ts

```
1  import { Component, OnInit } from '@angular/core';
2  import { Author} from '../author';
3  import {AuthorService} from '../author.service';
4
5  @Component({
6    selector: 'app-authors',
7    templateUrl: './authors.component.html',
8    styleUrls: ['./authors.component.css']
9  })
10 export class AuthorsComponent implements OnInit {
11   authors: Author[];
12
13   constructor(private authorService: AuthorService) { }
14
15   ngOnInit() {
16     this.getAuthors();
17   }
18
19   getAuthors(): void {
20     this.authorService.getAuthors().subscribe( next: authors => this.authors = authors);
21   }
22 }
```

Components

- Changes in “author_details.component.html”

author-details.component.html

```
1
2   <div *ngIf="author">
3     <h2>Information on {{ author.name | uppercase }} </h2>
4     <div>
5       <label>Num:
6         <input [ngModel]="author.id" readonly>
7       </label>
8     </div>
9     <div>
10      <label>Name:
11        <input [(ngModel)]="author.name" placeholder="name">
12      </label>
13    </div>
14    <div>
15      <label>Email:
16        <input [(ngModel)]="author.email" placeholder="email">
17      </label>
18    </div>
19    <button (click)="update()">Update</button>
20    <button (click)="delete()">Delete</button><br>
21    <button (click)="goBack()">go back</button>
22  </div>
```

Components

- Changes in “author_details.component.ts”

author-details.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Author} from '../author';
import { AuthorService} from '../author.service';
import { ActivatedRoute } from '@angular/router';
import { Location } from '@angular/common';

@Component({
  selector: 'app-author-details',
  templateUrl: './author-details.component.html',
  styleUrls: ['./author-details.component.css']
})
export class AuthorDetailsComponent implements OnInit {
  author: Author;

  constructor(
    private route: ActivatedRoute,
    private location: Location,
    private authorService: AuthorService
  ) {}

  ngOnInit() {
    this.getAuthor();
  }
```

Components

- Changes in “author_details.component.ts”

author-details.component.ts

```
25    getAuthor(): void {
26      const id = +this.route.snapshot.paramMap.get('id');
27      this.authorService.getAuthorId(id).subscribe( next: author => this.author = author);
28    }
29
30    update(): void {
31      this.authorService.updateAuthor(this.author).subscribe( next: () => this.goBack());
32    }
33
34    delete(): void {
35      this.authorService.deleteAuthor(this.author).subscribe( next: () => this.goBack());
36    }
37
38    goBack(): void {
39      this.location.back();
40    }
```