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 <u>Services</u> in Angular.

Services

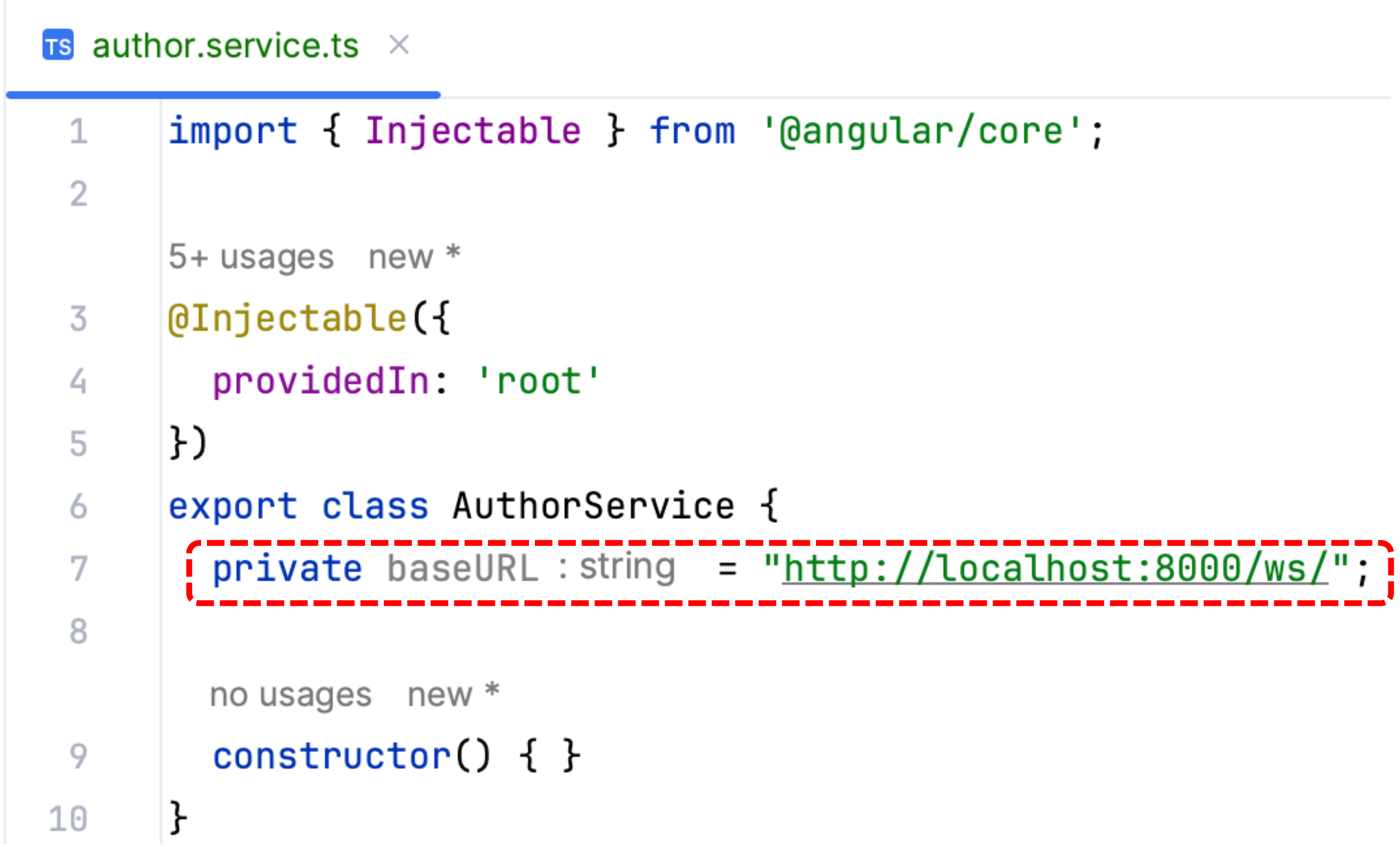
- A Service is a broad category encompassing any value, function, or feature that an app needs.
- A service is typically a class with a narrow, well-defined purpose. It should do something specific and do it well.
- By defining that kind of processing task in an injectable service class, you make it 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



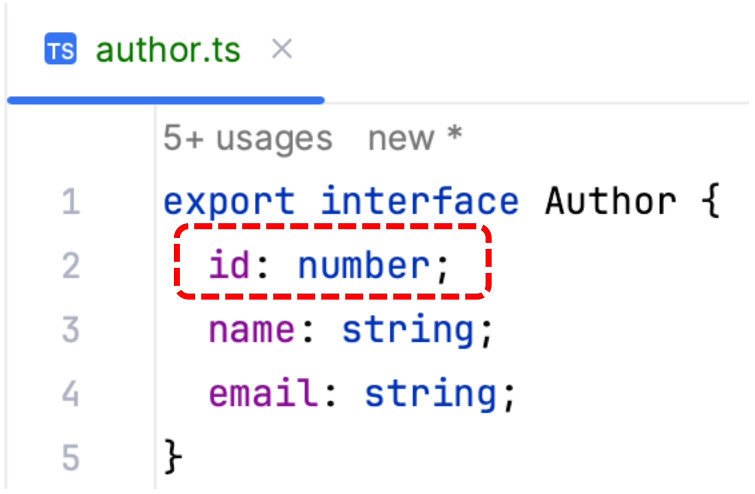
```
import { Injectable } from '@angular/core';

5+ usages   new *
@Injectable({
  providedIn: 'root'
})
export class AuthorService {
  private baseURL : string   = "http://localhost:8000/ws/";

  no usages   new *
  constructor() { }
}
```

Creating a Service (iii)

- Change Author interface as follows:

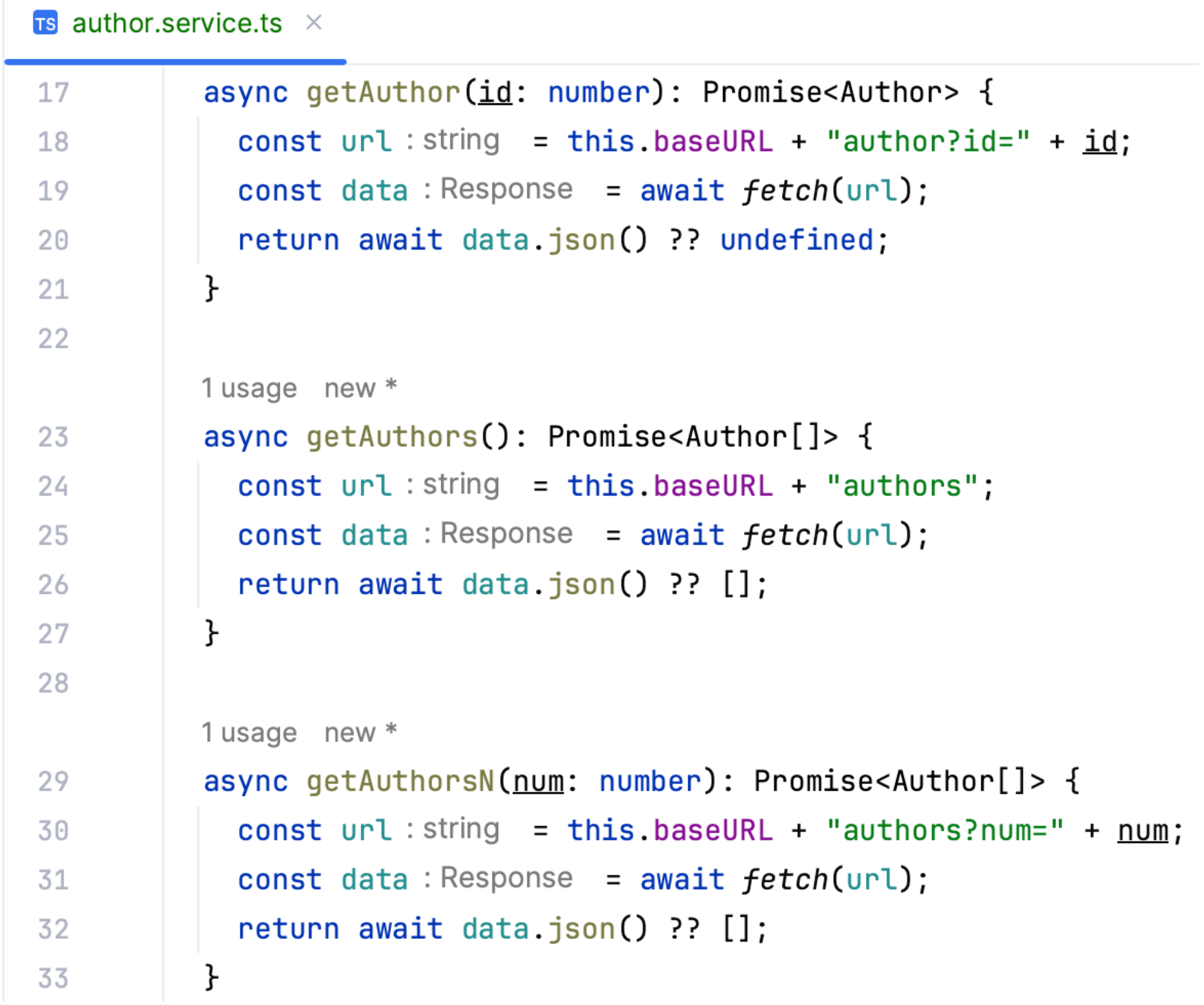


author.ts

```
5+ usages   new *
export interface Author {
  id: number;
  name: string;
  email: string;
}
```

Creating a Service (iv)

- Getting authors.



author.service.ts

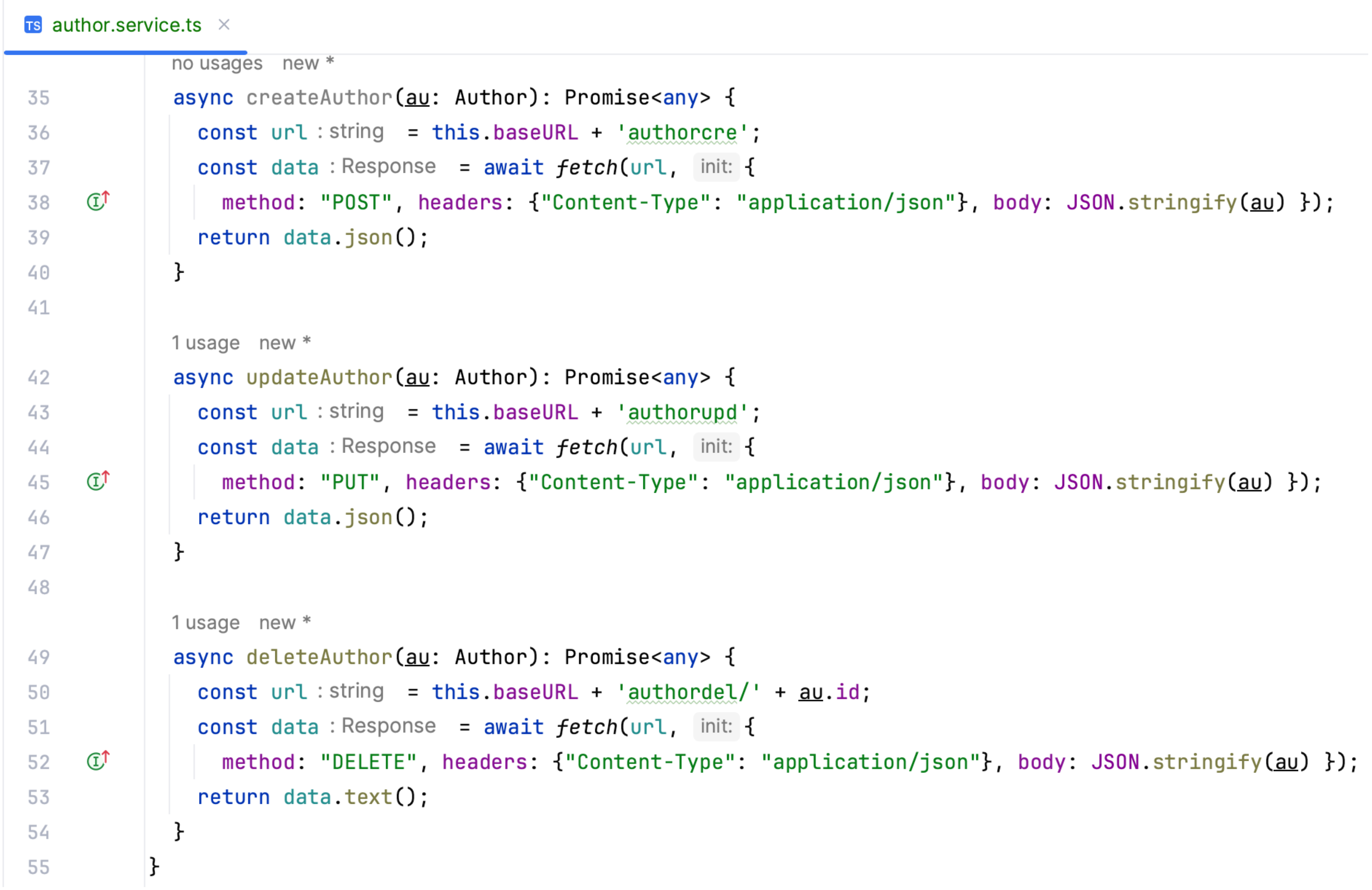
```
async getAuthor(id: number): Promise<Author> {
  const url :string  = this.baseURL + "author?id=" + id;
  const data :Response  = await fetch(url);
  return await data.json() ?? undefined;
}

1 usage   new *
async getAuthors(): Promise<Author[]> {
  const url :string  = this.baseURL + "authors";
  const data :Response  = await fetch(url);
  return await data.json() ?? [];
}

1 usage   new *
async getAuthorsN(num: number): Promise<Author[]> {
  const url :string  = this.baseURL + "authors?num=" + num;
  const data :Response  = await fetch(url);
  return await data.json() ?? [];
}
```

Creating a Service (v)

- Changing authors.



author.service.ts

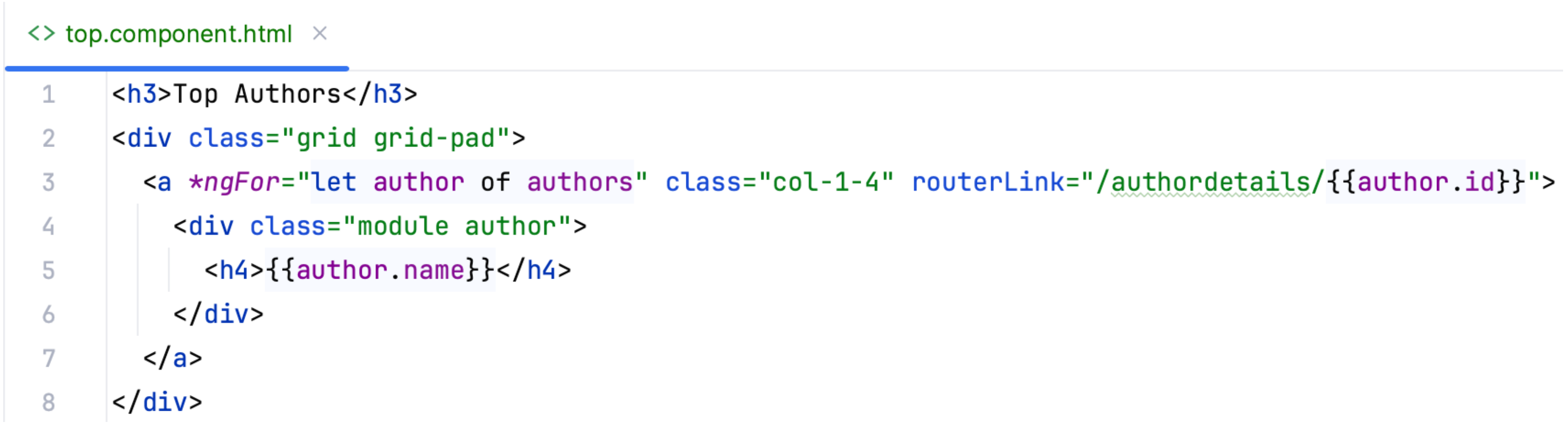
```
no usages   new *
async createAuthor(au: Author): Promise<any> {
  const url : string  = this.baseURL + 'authorcre';
  const data : Response  = await fetch(url,  init: {
    method: "POST", headers: {"Content-Type": "application/json"}, body: JSON.stringify(au) });
  return data.json();
}

1 usage   new *
async updateAuthor(au: Author): Promise<any> {
  const url : string  = this.baseURL + 'authorupd';
  const data : Response  = await fetch(url,  init: {
    method: "PUT", headers: {"Content-Type": "application/json"}, body: JSON.stringify(au) });
  return data.json();
}

1 usage   new *
async deleteAuthor(au: Author): Promise<any> {
  const url : string  = this.baseURL + 'authordel/' + au.id;
  const data : Response  = await fetch(url,  init: {
    method: "DELETE", headers: {"Content-Type": "application/json"}, body: JSON.stringify(au) });
  return data.text();
}
}
```

Components

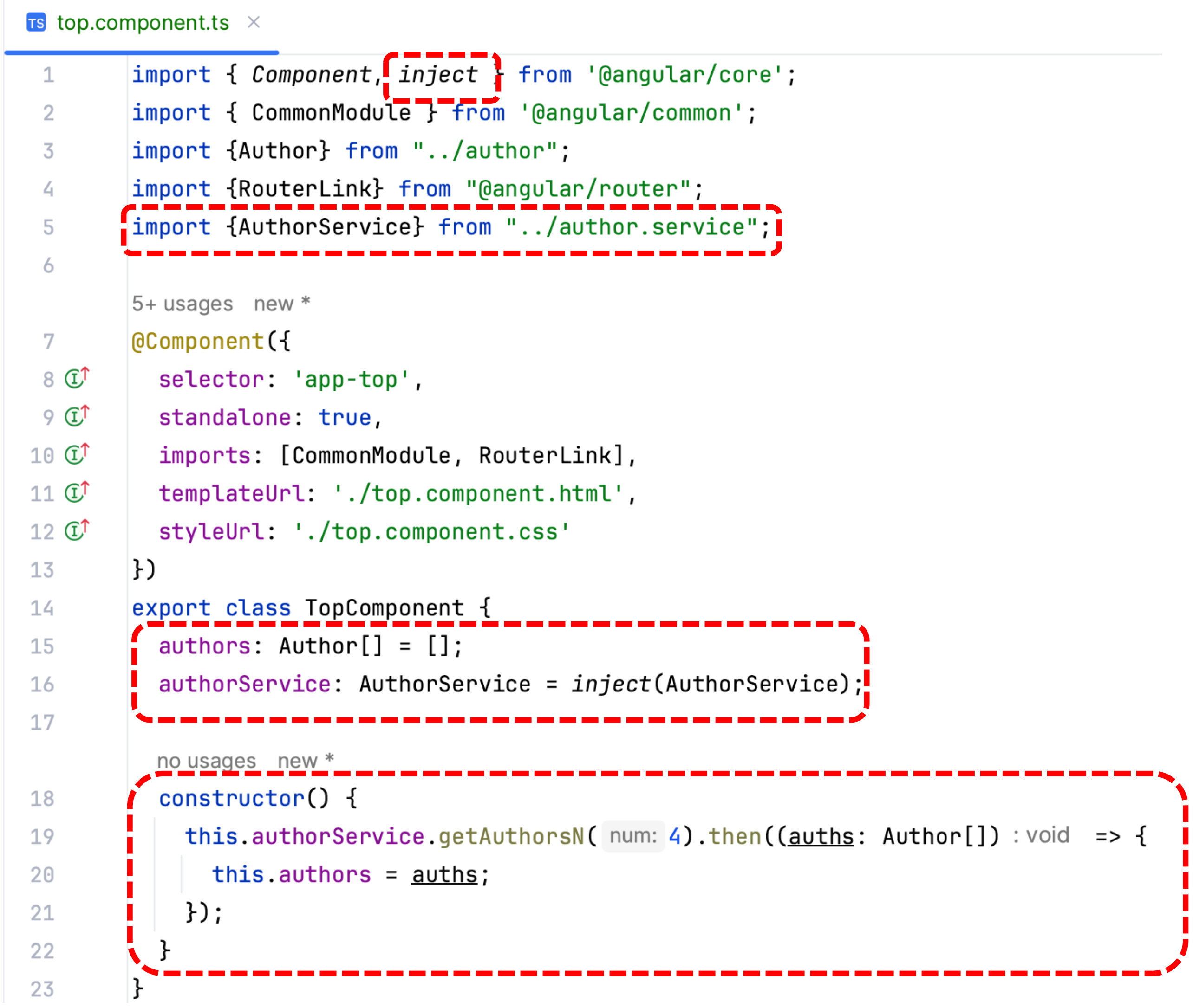
- Changes in “top.component.html”



```html
<h3>Top Authors</h3>
<div class="grid grid-pad">
  <a *ngFor="let author of authors" class="col-1-4" routerLink="/authordetails/{{author.id}}">
    <div class="module author">
      <h4>{{author.name}}</h4>
    </div>
  </a>
</div>
```

Components

- Changes in “top.component.ts”



top.component.ts

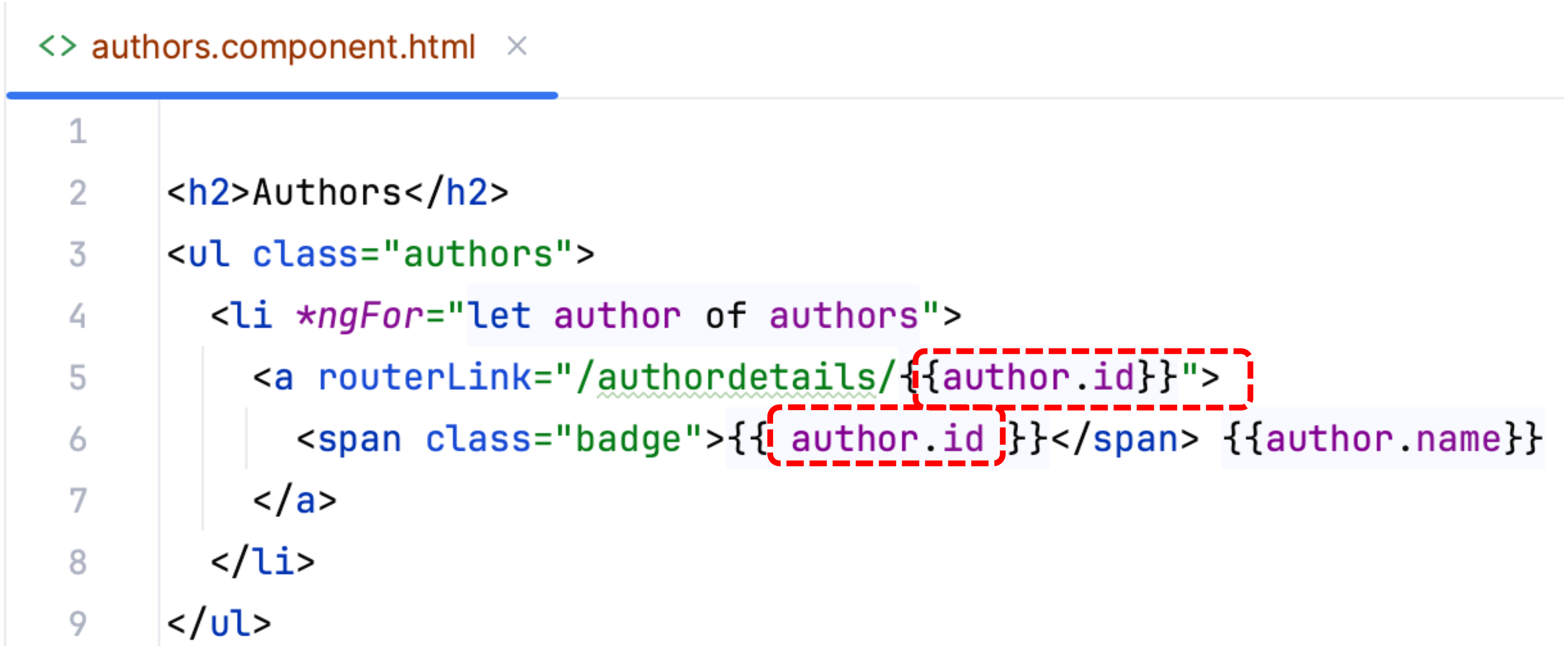
```
import { Component, inject } from '@angular/core';
import { CommonModule } from '@angular/common';
import {Author} from "../author";
import {RouterLink} from "@angular/router";
import {AuthorService} from "../author.service";

5+ usages   new *
@Component({
  selector: 'app-top',
  standalone: true,
  imports: [CommonModule, RouterLink],
  templateUrl: './top.component.html',
  styleUrl: './top.component.css'
})
export class TopComponent {
  authors: Author[] = [];
  authorService: AuthorService = inject(AuthorService);

  no usages   new *
  constructor() {
    this.authorService.getAuthorsN( num: 4).then((auths: Author[]) : void  => {
      this.authors = auths;
    });
  }
}
```

Components

- Changes in “authors.component.html”



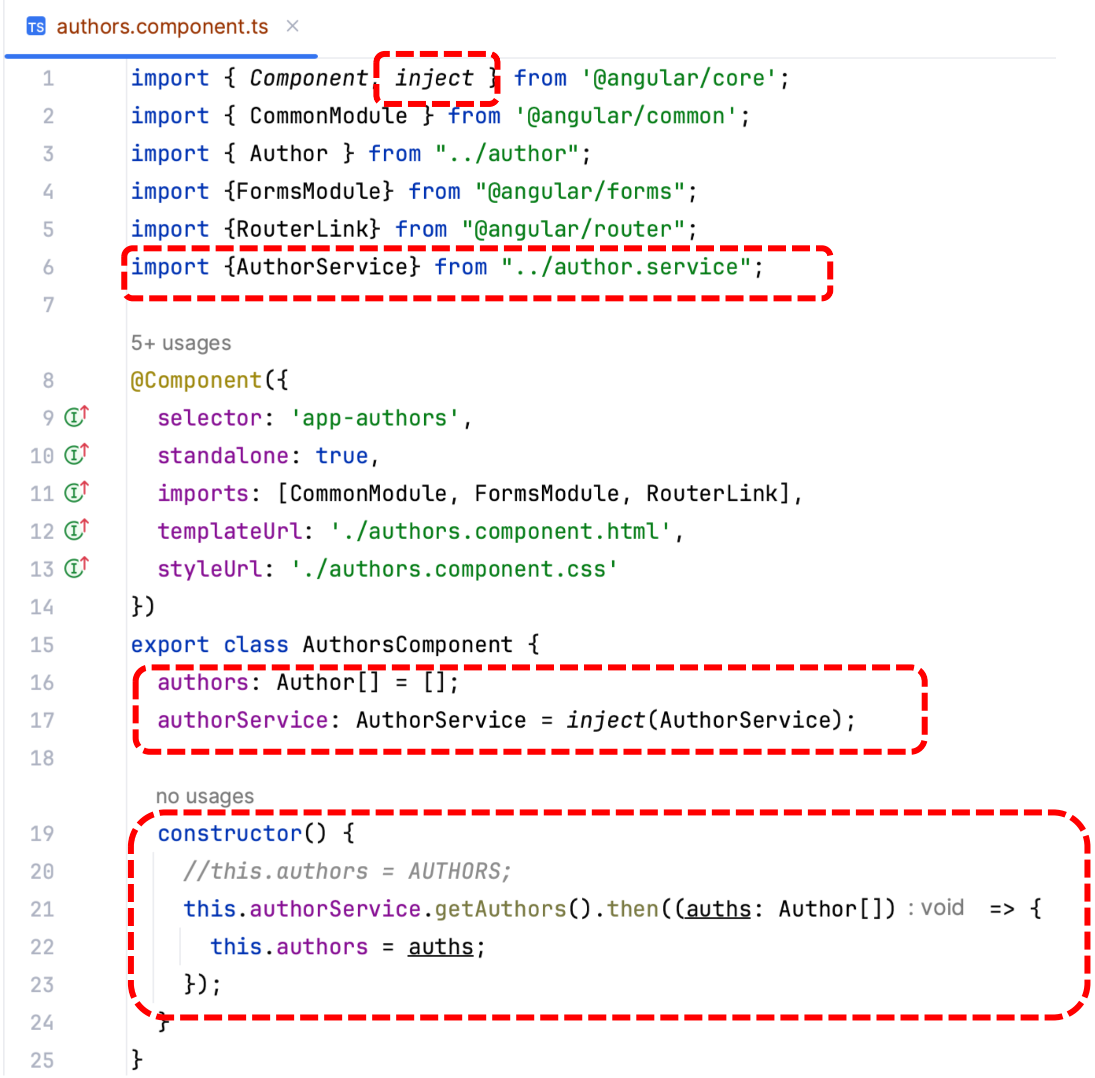
authors.component.html

```html

<h2>Authors</h2>
<ul class="authors">
  <li *ngFor="let author of authors">
    <a routerLink="/authordetails/{{author.id}}">
      <span class="badge">{{ author.id }}</span> {{author.name}}
    </a>
  </li>
</ul>
```

Components

- Changes in “authors.component.ts”



authors.component.ts

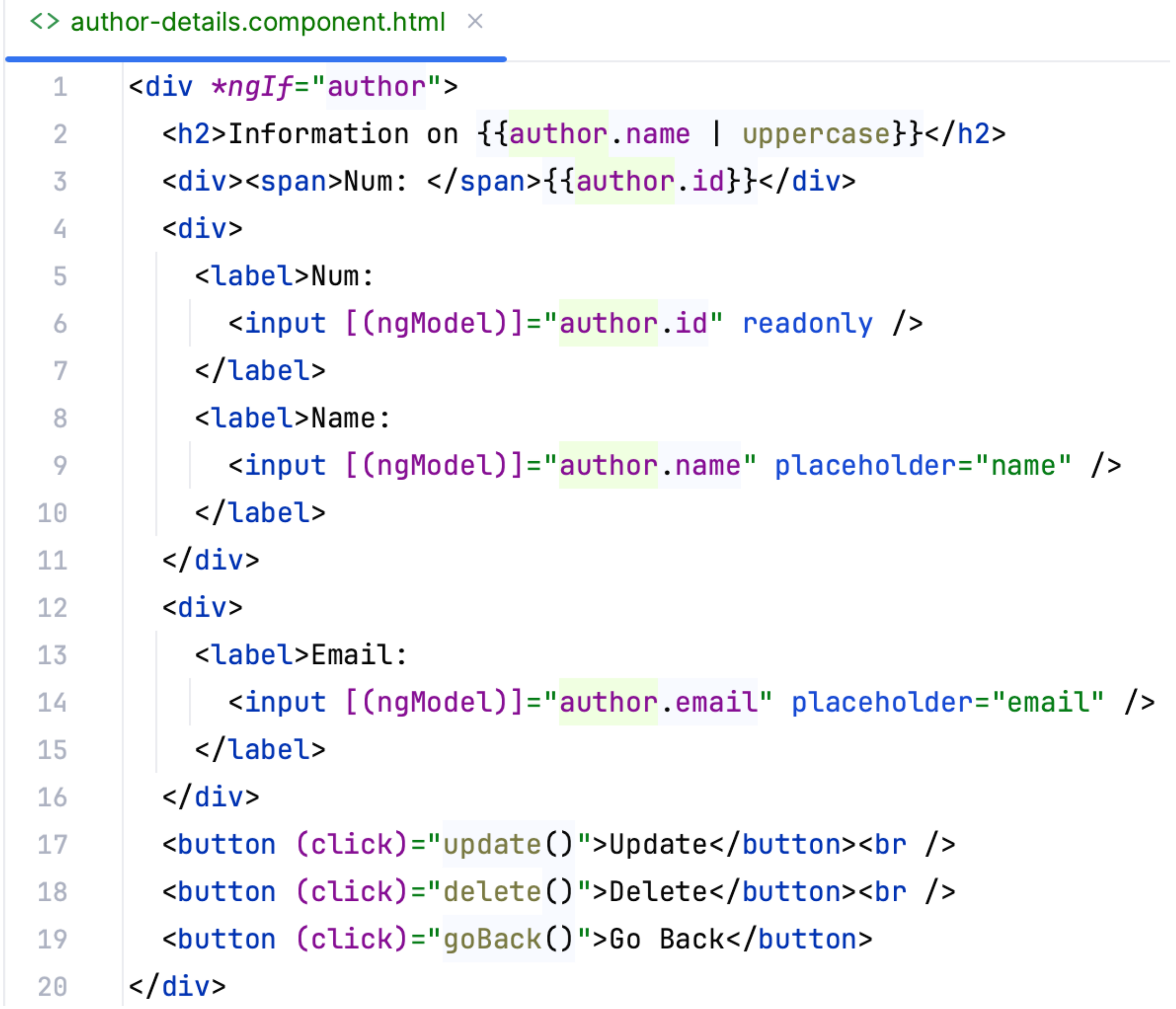
```
import { Component, inject } from '@angular/core';
import { CommonModule } from '@angular/common';
import { Author } from "../author";
import {FormsModule} from "@angular/forms";
import {RouterLink} from "@angular/router";
import {AuthorService} from "../author.service";

5+ usages
@Component({
  selector: 'app-authors',
  standalone: true,
  imports: [CommonModule, FormsModule, RouterLink],
  templateUrl: './authors.component.html',
  styleUrl: './authors.component.css'
})
export class AuthorsComponent {
  authors: Author[] = [];
  authorService: AuthorService = inject(AuthorService);

  no usages
  constructor() {
    //this.authors = AUTHORS;
    this.authorService.getAuthors().then((auths: Author[]) : void  => {
      this.authors = auths;
    });
  }
}
```

Components

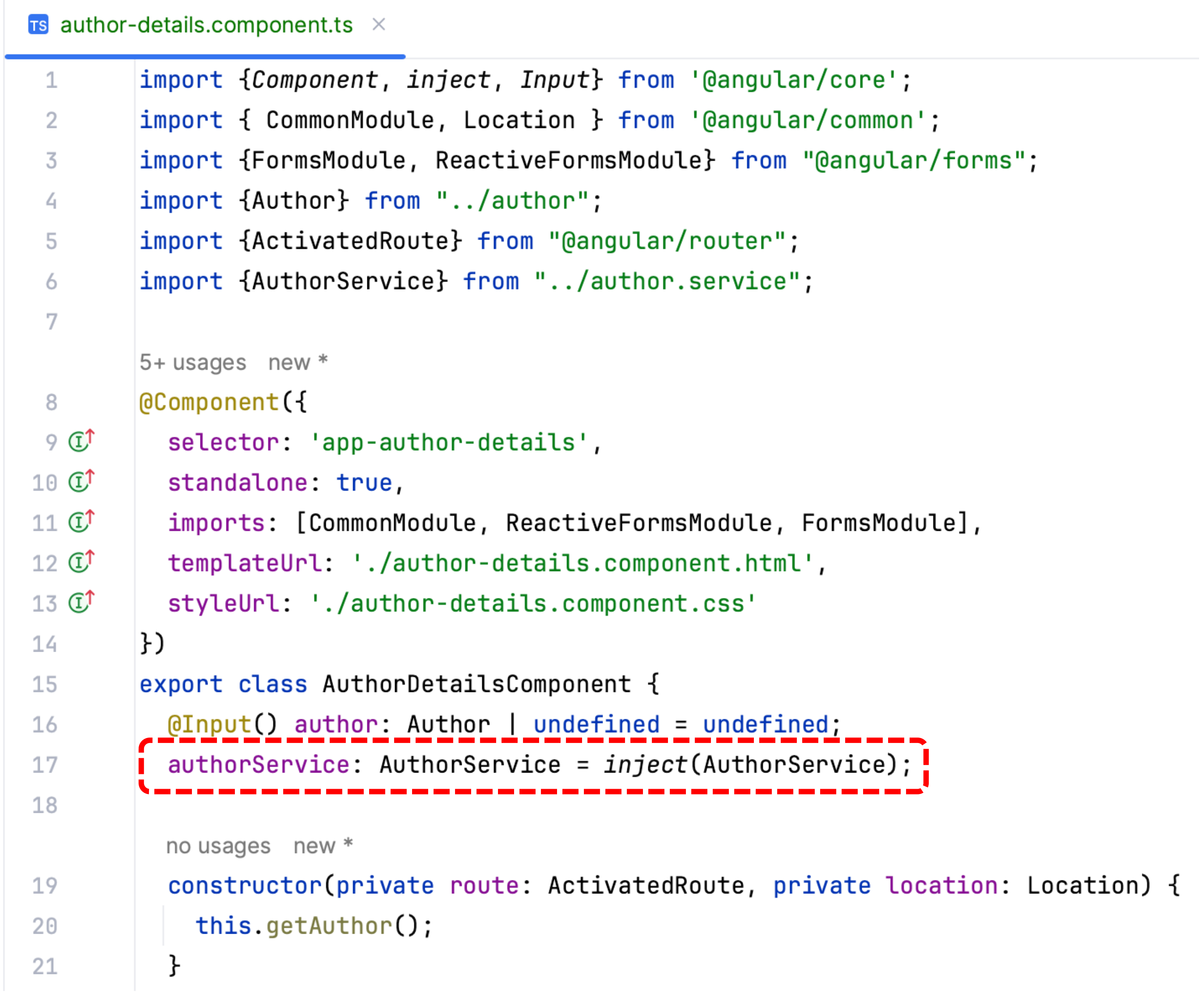
- Changes in “author_details.component.html”


author-details.component.html

```html
<div *ngIf="author">
  <h2>Information on {{author.name | uppercase}}</h2>
  <div><span>Num: </span>{{author.id}}</div>
  <div>
    <label>Num:
      <input [(ngModel)]="author.id" readonly />
    </label>
    <label>Name:
      <input [(ngModel)]="author.name" placeholder="name" />
    </label>
  </div>
  <div>
    <label>Email:
      <input [(ngModel)]="author.email" placeholder="email" />
    </label>
  </div>
  <button (click)="update()">Update</button><br />
  <button (click)="delete()">Delete</button><br />
  <button (click)="goBack()">Go Back</button>
</div>
```

Components

- Changes in "author details.component.ts"



author-details.component.ts

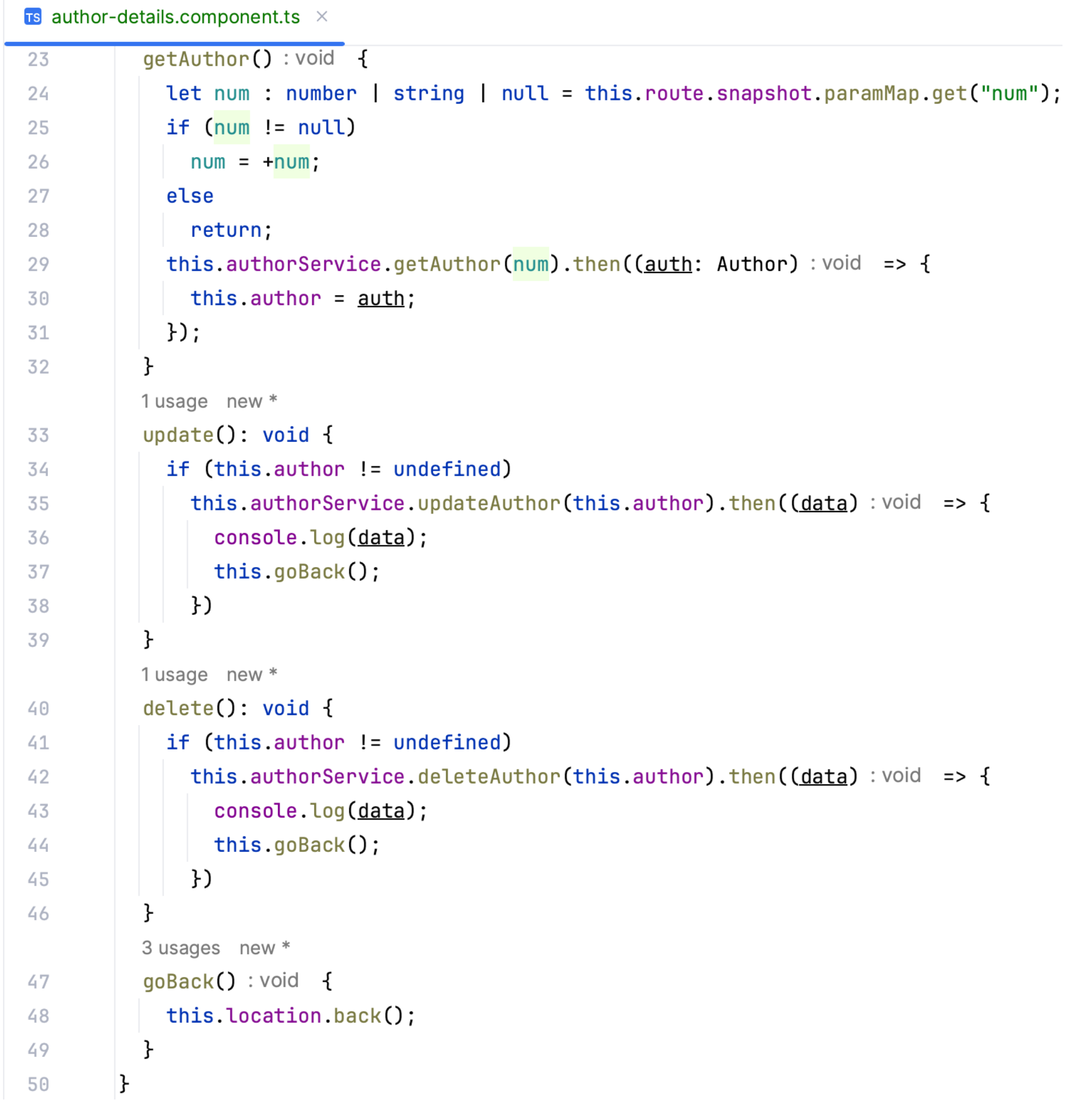
```
import {Component, inject, Input} from '@angular/core';
import { CommonModule, Location } from '@angular/common';
import {FormsModule, ReactiveFormsModule} from "@angular/forms";
import {Author} from "../author";
import {ActivatedRoute} from "@angular/router";
import {AuthorService} from "../author.service";

5+ usages   new *
@Component({
  selector: 'app-author-details',
  standalone: true,
  imports: [CommonModule, ReactiveFormsModule, FormsModule],
  templateUrl: './author-details.component.html',
  styleUrl: './author-details.component.css'
})
export class AuthorDetailsComponent {
  @Input() author: Author | undefined = undefined;
  authorService: AuthorService = inject(AuthorService);

  no usages   new *
  constructor(private route: ActivatedRoute, private location: Location) {
    this.getAuthor();
  }
```

Components

- Changes in “author_details.component.ts”



author-details.component.ts

```
getAuthor() : void  {
  let num : number | string | null = this.route.snapshot.paramMap.get("num");
  if (num != null)
    num = +num;
  else
    return;
  this.authorService.getAuthor(num).then((auth: Author) : void  => {
    this.author = auth;
  });
}
1 usage   new *
update(): void {
  if (this.author != undefined)
    this.authorService.updateAuthor(this.author).then((data) : void  => {
      console.log(data);
      this.goBack();
    })
}
1 usage   new *
delete(): void {
  if (this.author != undefined)
    this.authorService.deleteAuthor(this.author).then((data) : void  => {
      console.log(data);
      this.goBack();
    })
}
3 usages   new *
goBack() : void  {
  this.location.back();
}
}
```