Gyakorlati feladatlap

Angular alapok + HTTP/Service/Observable

Téma: Mini “User Directory” alkalmazás

Időtartam: 2 × 45 perc

Cél: komponensek, binding, routing, service, dependency injection, HttpClient, Observable, error handling, interceptor alapok

Tanulási célok

- Megérteni a Component–Template–Binding kapcsolatot (UI állapot és események).
- Megérteni, miért Service-ben van az adatlekérés (felelősségszétválasztás).
- HttpClient használata: GET lista + GET részletek.
- Observable szemlélet: a HTTP kérés subscribe-nál indul, next/error/complete.
- Egyszerű hibakezelés: a Service “értelmezi”, a Komponens csak megjelenít.
- Interceptor szerepe: közös header + alap logolás minden kérésnél.

Előfeltételek

- Node.js és npm telepítve.
- Angular CLI telepítve (ng parancs működik).
- Szerkesztő: VS Code ajánlott.
- Internet elérés a publikus teszt API-hoz.

Használt teszt API: https://jsonplaceholder.typicode.com/users

Fontos szabályok a gyakorlat során

- A Komponens a UI-ért felel: megjelenítés + gombkattintás kezelése.
- A Service felel az adatokért: HTTP hívás és hibakezelés.
- A Komponens nem hív közvetlenül HttpClient-et.
- Hibánál a Komponens csak egy hibaüzenetet jelenít meg, és ad “Újrapróbál” gombot.

1. rész: Komponensek, Binding, Routing

1.1 Projekt létrehozása

Hozd létre az Angular projektet routinggal és module támogatással:

```
ng new user-directory --routing --style=css --standalone=false
cd user-directory
ng serve -o
```

Megjegyzés: Az Angular CLI 17+ verziók alapértelmezésben standalone alkalmazást hoznak létre. Ebben a gyakorlatban KIFEJEZETTEN a klasszikus és elterjedt AppModule-alapú struktúrát használjuk.

- Ellenőrzőpontok:
  - A böngészőben megjelenik az alap Angular oldal.
  - A terminálban nincs build hiba.

1.2 Komponensek legenerálása

Generáld le a következő komponenseket:

```
ng generate component pages/user-list
ng generate component pages/user-details
ng generate component shared/navbar
```

Megjegyzés: A pages/ mappába kerülnek az “oldalak”, a shared/ alá a közös UI elemek (pl. menü).

1.3 Navbar egyszerű menü (routerLink)

A shared/navbar komponens template-jében készíts egyszerű navigációt:

```
<nav>
  <a routerLink="/users">Users</a>
</nav>
<hr />
```

Tedd be az app komponens template-jébe a navbart a router-outlet fölé.

```
<app-navbar></app-navbar>
<router-outlet></router-outlet>
```

1.4 Routing beállítása

Az app routingban (AppRoutingModule) állítsd be az útvonalakat:

```
const routes: Routes = [
  { path: '', pathMatch: 'full', redirectTo: 'users' },
  { path: 'users', component: UserListComponent },
  { path: 'users/:id', component: UserDetailsComponent },
  { path: '**', redirectTo: 'users' }
];
```

- Ellenőrzőpontok:
  - A /users megnyitásakor a UserList oldal jelenik meg.
  - Ismeretlen útvonalon (pl. /asdf) visszadob /users-re.

1.5 Binding gyakorlat (egyszerű keresőmező)

A UserList komponensben készíts keresőmezőt és jelenítsd meg az aktuális értéket.

1) A komponens osztályban legyen egy változó:

```
searchText: string = '';
```

2) A template-ben használd a two-way bindingot (ngModel).

```
<label>
  Keresés:
  <input [(ngModel)]="searchText" placeholder="pl. Leanne" />
</label>
<p>Keresés: <strong>{{ searchText }}</strong></p>
```

Ehhez importáld a FormsModule-t az AppModule-ba.

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

@NgModule({
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule
]
})
```

- Ellenőrzőpontok:
  - Gépelés közben frissül a “Keresés:” érték.
  - Nincs konzol hiba az ngModel miatt.

2. rész: Service, HTTP, Observable, Error, Interceptor

2.1 Model (User interface) létrehozása

Hozz létre egy egyszerű típust a users adatokhoz:

```
// src/app/core/user.model.ts
export interface User {
  id: number;
  name: string;
  email: string;
  phone: string;
}
```

2.2 UserService generálása

Generáld le a service-t:

```
ng generate service core/user
```

A service feladata: HTTP hívások és a hibák egységes kezelése.

2.3 HttpClientModule bekötése

Az AppModule-ban importáld a HttpClientModule-t:

```
import { HttpClientModule } from '@angular/common/http';

@NgModule({
  imports: [BrowserModule, AppRoutingModule, FormsModule, HttpClientModule]
})
```

2.4 HTTP metódusok a service-ben (Observable)

Készíts két metódust: getUsers() és getUser(id). A visszatérés Observable legyen.

```
// src/app/core/user.service.ts
import { Injectable } from '@angular/core';
import { HttpClient, HttpErrorResponse } from '@angular/common/http';
import { Observable, throwError } from 'rxjs';
import { catchError } from 'rxjs/operators';
import { User } from './user.model';

@Injectable({
  providedIn: 'root'
})
export class UserService {
  private readonly baseUrl = 'https://jsonplaceholder.typicode.com/users';

  constructor(private http: HttpClient) {}

  getUsers(): Observable<User[]> {
    return this.http.get<User[]>(this.baseUrl).pipe(
      catchError(err => this.handleError(err))
    );
  }
```

```
  getUser(id: number): Observable<User> {
    return this.http.get<User>(`${this.baseUrl}/${id}`).pipe(
      catchError(err => this.handleError(err))
    );
  }

  private handleError(err: unknown) {
    // Egyszeru, oktatasi cel:
    if (err instanceof HttpErrorResponse) {
      if (err.status === 0) {
        return throwError(() => new Error('Halozati hiba: nem elerheto a
szerver.'));
      }
      if (err.status === 404) {
        return throwError(() => new Error('Nem talalhato (404).'));
      }
      return throwError(() => new Error(`Backend hiba: ${err.status}`));
    }
    return throwError(() => new Error('Ismeretlen hiba.'));
  }
}
```

Megjegyzés: Figyeld meg: a Service alakítja egységes hibaüzenetté a problémát. A Komponens csak megjeleníti.

2.5 UserList: adatok betöltése (subscribe, loading, error)

A UserList komponensben kezeld a 3 állapotot: betöltés, hiba, siker.

```
// user-list.component.ts (vaz)
users: User[] = [];
filteredUsers: User[] = [];
searchText: string = '';
isLoading: boolean = false;
errorMessage: string | null = null;

constructor(private userService: UserService) {}

ngOnInit() {
  this.loadUsers();
}

loadUsers() {
  this.isLoading = true;
  this.errorMessage = null;

  this.userService.getUsers().subscribe({
    next: (data) => {
      this.users = data;
      this.applyFilter();
    },
    error: (err: Error) => {
      this.errorMessage = err.message;
    },
    complete: () => {
      this.isLoading = false;
    }
  });
}

applyFilter() {
  const q = this.searchText.toLowerCase().trim();
  this.filteredUsers = this.users.filter(u =>
    u.name.toLowerCase().includes(q)
  );
}
```

```
<h2>Users</h2>

<label>
  Keresés:
  <input [(ngModel)]="searchText" (ngModelChange)="applyFilter()" placeholder="pl.
Leanne" />
</label>

<div *ngIf="isLoading">Loading...</div>

<div *ngIf="errorMessage" style="color: #b00020;">
  <p><strong>Hiba:</strong> {{ errorMessage }}</p>
```

```
  <button (click)="loadUsers()">Ujrapróbál</button>
</div>

<ul *ngIf="!isLoading && !errorMessage">
  <li *ngFor="let u of filteredUsers">
    <a [routerLink]="['/users', u.id]">{{ u.name }}</a>
    <small>({{ u.email }})</small>
  </li>
</ul>
```

- Ellenőrzőpontok:
  - A lista betöltődik es megjelenik.
  - Gépelésre szűr a lista (egyszerű filter).
  - Hiba esetén megjelenik az üzenet + Újrapróbál gomb.

2.6 UserDetails: route param + service hívás

Olvasd ki az id-t az URL-ből és kérd le a user részleteit.

```
// user-details.component.ts (vaz)
user: User | null = null;
isLoading: boolean = false;
errorMessage: string | null = null;

constructor(
  private route: ActivatedRoute,
  private userService: UserService
) {}

ngOnInit() {
  const id = Number(this.route.snapshot.paramMap.get('id'));
  this.isLoading = true;
  this.userService.getUser(id).subscribe({
    next: (u) => this.user = u,
    error: (err: Error) => this.errorMessage = err.message,
    complete: () => this.isLoading = false
  });
}
```

```
<a routerLink="/users">← Vissza</a>

<div *ngIf="isLoading">Loading...</div>

<div *ngIf="errorMessage" style="color: #b00020;">
  <p><strong>Hiba:</strong> {{ errorMessage }}</p>
</div>

<div *ngIf="!isLoading && user">
  <h2>{{ user.name }}</h2>
  <p>Email: {{ user.email }}</p>
  <p>Phone: {{ user.phone }}</p>
</div>
```

2.7 Interceptor: közös header + log

Generáld le az interceptort:

```
ng generate interceptor core/demo-auth
```

Feladat: minden kéréshez add hozzá a X-Demo-Token fejlécet, és logold a kérést a konzolra.

```
// demo-auth.interceptor.ts (vaz)
import { Injectable } from '@angular/core';
import {
  HttpInterceptor, HttpRequest, HttpHandler, HttpEvent
} from '@angular/common/http';
import { Observable, tap } from 'rxjs';

@Injectable()
export class DemoAuthInterceptor implements HttpInterceptor {
  intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
    const cloned = req.clone({
      setHeaders: {
        'X-Demo-Token': '12345'
      }
    });

    console.log('[HTTP] Request:', cloned.method, cloned.url);

    return next.handle(cloned).pipe(
      tap({
        error: (err) => console.log('[HTTP] Error:', cloned.url, err?.status),
        complete: () => console.log('[HTTP] Complete:', cloned.url)
      })
    );
  }
}
```

Megjegyzés: Interceptor: közös szabályok egy helyen (auth header, logolas, globális hibakezelés).

Befejezés - Mit kell tudnod a végére?

- A **Komponens** csak UI: betöltés/hiba/adat megjelenítés.
- A **Service** végzi a HTTP hívásokat és a hibak egységesiteset.
- A HttpClient **Observable**-t ad: a HTTP kérés subscribe-nál indul, next/error/complete kezeli az életciklust.
- Az **Interceptor** minden kerest erint: kozos header + alap logolas.

Opcionális: Adj hozzá egy „**Retry**” gombot a *Details* oldalra is, vagy jeleníts meg egy egyszerű „Nincs találat” üzenetet, ha a filter üres listát ad.