ANGULAR

Segédlet a Kliens oldali technológiák c. tárgyhoz

Szabó Gábor

2020.

Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Automatizálási és Alkalmazott Informatikai Tanszék

1117 Budapest, Magyar tudósok körútja 2.
Q. ép. B. szárny 207. • www.aut.bme.hu
Telefon: 463-2870 • Fax: 463-2871

Bevezetés

Célkitűzés

A labor során TypeScript alapismereteinket felhasználva megismerkedünk az Angular alkalmazás-fejlesztő keretrendszer alapszintű használatával.

Előfeltételek

A labor elvégzéséhez szükséges eszközök:

- **Visual Studio Code** (tesztelve a 1.10.2 verzióval, a szerkesztő ezen verziója a tárgyi honlapról letölthető)
    - A VS Code ZIP verziója a hivalatos oldalról is letölthető, ekkor nem szükséges adminisztrátori jog a telepítéshez (http://code.visualstudio.com/docs/?dv=winzip)
- NodeJS Package Manager (**NPM**) (tesztelve NodeJS 6.10.0, NPM 3.10.10 verziókkal)
- **05 - Angular kiinduló** zip fájl vagy *05 - Angular kiinduló - függőségekkel* zip fájl a tanszéki portálról
    - Az *Angular kiinduló függőségekkel zip* tartalmazza az összes NPM függőséget is, a mérete így nagyobb, kb. 30 MB

Amit érdemes átnézned

- HTML, JavaScript alapok
- TypeScript típusrendszer, statikus típusosság
- Angular architektúra, adatkötés alapok

Lebonyolítás

A gyakorlat anyagát távolléti oktatás esetén önállóan, jelenléti oktatás esetében számítógépes laborban, gyakorlatvezetői útmutatással kell megoldani.

Beadás

Amennyiben elkészültél a megoldással, távolíts el belőle a fordítási binárisokat és fordítási segédmappákat („bin” mappa, „obj” mappa, „Visual Studio” mappa, esetleges nuget/node csomagok, majd az így elkészült anyagot egy zip fájlba becsomagolva töltsd fel a Moodle rendszerbe. Amennyiben a zip fájl mérete 10 MB fölött van, valószínűleg nem töröltél ki belőle mindent a korábbi felsorolásból. Jegyzőkönyv készítése nem szükséges, azonban amennyiben beadott kódoddal kapcsolatban kérdések merülnek fel, megkérhetünk arra, hogy működésének egyes részleteit utólag is magyarázd el.

Összefoglalás

Az előadásokat követően gyakorlatban is megismerkedünk az Angular alapú fejlesztéssel, amiben egy névjegykártya-kezelő alkalmazást készítünk el.

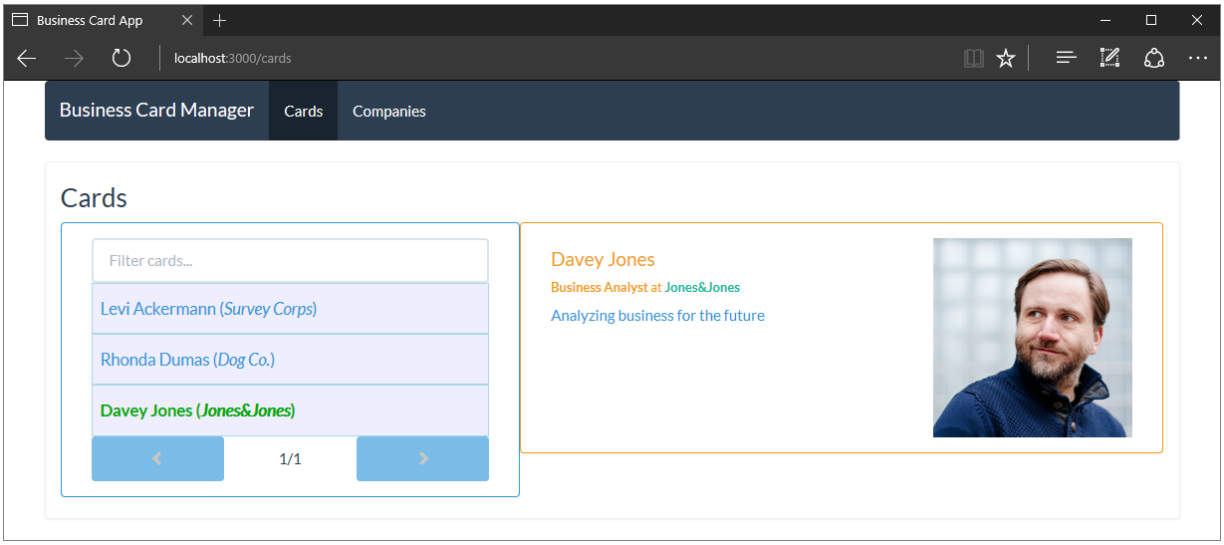
FELADATOK

0. FELADAT – ÁTTEKINTÉS

Első lépésként tekintsük át a kiinduló projektet, végezzük el a szükséges műveleteket a függőségek betöltéséhez, és indítsuk el az alkalmazást!

1. Ez a labor kódintenzív, ezért ha úgy véljük, nincs elég idő a kód bepötyögésére, használhatjuk az alábbi linken elérhető snippeteket: http://bit.ly/kliensoldalitech-angular
2. Nyissuk meg a Visual Studio Code szerkesztőt!
    - A szerkesztőben az **F1** megnyomására egy beépített, mindenre alkalmas keresőt és feladat-futtatót kapunk. Az itt megjelenő „>" jelet törölve és a „?" karaktert beírva gyors segítséget kapunk az egyes funkciók elvégzésére. A „**>**" után parancsokat gépelhetünk. A színséma átállításához begépelhetjük a „**>**" után a „**theme**" kulcsszót, majd a **Preferences: Color Theme**-et választva válthatunk a beépített színsémák között.
3. A File → Open Folder… paranccsal válasszuk a kiinduló projekt mappáját, ahová kitömörítettük a ZIP fájl tartalmát! Az alábbiakat érdemes áttekintenünk:
    - Az alkalmazás gyökerében és .vscode mappájában különböző .json kiterjesztésű konfigurációs fájlok találhatók.
    - A package.json fájlban az alkalmazásunk NPM függőségei vannak felsorolva, köztük az Angular keretrendszer, a TypeScript fordító, és a fejlesztésben sokat segítő lite-server webszerver és BrowserSync eszköz.
    - A függőségeket vagy az *Angular-kiindulo-fuggosegekkel.zip* fájlból csomagoljuk a megfelelő helyre vagy a **Ctrl+ö** billentyűkombinációval nyissuk fel a Terminal ablakot, ahová gépeljük be az alábbi parancsot: **npm install**! Ez a parancs NPM-ből letölti az összes csomagot, amire szükségünk van a **node_modules** mappába.
4. A függőségek letöltése/kitömörítése után a **Ctrl+Shift+B** billentyűkombinációval egyszerre elindul a *TypeScript compiler* watch módban, illetve a *lite-server* webkiszolgáló, ill. az alapértelmezett böngésző. A böngészőre ilyenkor felcsatlakozik a **BrowserSync**, ami a fájlrendszer adott mappájában bekövetkező változásokra (fájl mentése, új fájl létrehozása stb.) **automatikusan frissíti** az összes kapcsolódó böngészőablakot (hot reload). Tehát az első buildelés után többet elvileg nem kell semmit indítanunk, a kód automatikusan fordulni fog és ez a böngészőben automatikusan reflektálódik.
    - Ha mégis, bármilyen okból kifolyólag újra kell indítanunk a folyamatot, akkor az **F1 → > Tasks: Terminate Running Task** opcióval állítsuk le a szervert és a compilert, majd a Ctrl+Shift+B-vel indítsuk újra!
5. Ha telepítve van a Debugger for Chrome bővítmény, akkor az F5 hatására a Chrome böngészőben dolgozhatunk, és a hibakeresés a VS Code-ban működik (tehát VS Code-ban helyezhetünk el breakpointokat, vizsgálhatunk változókat stb.).
6. Az **index.html**-ben az alkalmazás indulásához szükséges HTML váz található, itt észrevehetjük a **business-card-app** HTML elemet, ami az alkalmazásunk bootstrap-elt komponense lesz. A main.ts-ben a **BusinessCardAppModule**-t indítjuk el, amin F12-t nyomhatunk és láthatjuk, hogy hogyan van definiálva a kiinduló modul:
    - az útvonalválasztás megfelelően van konfigurálva, a **/companies**, **/cards**, **/companies/:id** és **/cards/:id** útvonalakat fogjuk használni, ami a megfelelő komponenst fogja kiválasztani, és a **BusinessCardAppComponent** HTML template-jében található **<router-outlet>** direktívával kezelni.

7. Az alkalmazás kódja az app mappában található, az egyetlen modulon kívül 3 komponens (egy teljes oldalt körülölelő komponens és két saját oldalt reprezentáló komponens) található a komponens kódjával és template-jével, valamint 3 modell osztály és 2 szolgáltatás. Az alkalmazás 2 fontos üzleti entitása a névjegykártya (**Card** interfész) és a cég (**Company** interfész), ezeket a **models** mappában vizsgáljuk meg!
8. Ha elindítjuk a http://localhost:3000/cards URL-en vagy a Cards menüpontra kattintva az alábbi képernyő fogad:



1. FELADAT – ADATKÖTÉS A CARDS OLDALON

Láthatjuk, hogy a jelenleg kitöltött adatok az oldalon statikusak, nem használjuk a felelősségek szétválasztása elvet az adatok megszerzésére és betöltésére, valamint nem használunk semmilyen adatkötést az adatok megjelenítéséhez. Ezen változtassunk!

A BusinessCardAppComponent osztályban kérjük el a CardService által visszaadott kártyák listáját, és jelenítsük meg azokat a felületen!

---

*Tipp: kódot formázni a **Shift+Alt+F** billentyűkombinációval tudunk. A hiányzó hivatkozások betöltéséhez használjuk a **Ctrl+.** billentyűkombinációt a hiányzó deklaráción! További billentyűkombinációkért az F1 → Preferences: Open Keyboard Shortcuts lehetőséget keressük fel!*

---

app/components/card-page/card-page.component.ts:

```
export class CardPageComponent implements OnInit {
    constructor(private cardService: CardService) { }
    ngOnInit() {
        this.getCards();
    }
    getCards() {
        this.cards = this.cardService.getCards().map(r => r.results);
    }
    selectedCard: Card;
    cards: Observable<Card[]>;
}
```

Tipp: az OnInit megvalósításban megírt kódot sokszor alapból érdemes kiszerveznünk egy-egy saját függvénybe, mert ezeket később könnyebb lesz eseményekkel adatkötnünk.

Tehát a komponens az inicializációt követően (**ngOnInit** metódus) a **cards** tulajdonság értékét feltölti: az injektált **CardService** példánytól elkéri egy aszinkron kérés eredményét. A jelenlegi implementációja csak mock adatokkal dolgozik lokálisan, viszont a komponensnek ez transzparens, a háttérben ugyanígy használhatnánk egy olyan szolgáltatást is, ami valójában HTTP-n kommunikál.

A felületet ennek megfelelően módosítanunk kell.

app/components/card-page/card-page.component.html:

```
<!-- Here comes the list of cards. -->
<div class="cards-list">
    <a *ngFor="let card of cards | async"
       [class.active]="card === selectedCard">
        <h6 class="text-info">{{card.firstName}} {{card.lastName}}
            (<i>{{card.companyId}}</i>)</h6>
    </a>
</div>
```

Módosítás és mentés után a felület automatikusan frissül és immáron a szolgáltatás által szolgáltatott adatokat kötjük ki a felületre.

Láthatjuk, hogy a lapozást nem tudjuk megoldani, csak ha a megfelelő `SearchResult<Card>` Observable-t használjuk fel, ugyanis abban találhatók meg a lapozáshoz szükséges adatok. Módosítsuk megfelelően a komponenst és a template-et:

app/components/card-page/card-page.component.ts:

```
export class CardPageComponent implements OnInit {
    getCards() {
        this.cards = this.cardService.getCards().map(r => r.results);
    }
    cards: Observable<SearchResult<Card>>;
}
```

app/components/card-page/card-page.component.html:

```
<a *ngFor="let card of (cards | async)?.results"
   [class.active]="card === selectedCard">
```

A kártyára történő kattintáskor pedig válasszuk ki az adott kártyát, és annak a részleteit jelenítsük meg a jobb oldalon:

app/components/card-page/card-page.component.html:

```
<a *ngFor="let card of (cards | async)?.results" [class.active]="card === sele
ctedCard" (click)="selectedCard = card">
```

```
<div class="col-lg-7 panel panel-warning current-card" *ngIf="selectedCard">
    <!-- Here comes the selected or newly created card's editor / detail. -->
    <div class="panel-body">
```

```
        <img class="profile-picture pull-right" [src]="selectedCard.imageUrl">
        <h4 class="text-warning">
            {{selectedCard.firstName}} {{selectedCard.lastName}}</h4>
        <h6 class="text-warning"><b>{{selectedCard.role}}</b>
            at <b><a>{{selectedCard.companyId}}</a></b></h6>
        <p class="text-info">{{selectedCard.motto}}</p>
    </div>
</div>
```

A felület frissül, és láthatjuk, hogy megfelelően ki tudjuk választani az aktuális elemet, aminek a jobb oldalon megjelennek a részletei.

A cég ID-ja helyett viszont sokkal érdemesebb magát a cégnevet kikötnünk a felületre. Ehhez a CompanyService-t használva kérjük el (és tároljuk el) az aktuális cégeket, és ID alapján kérjük el annak a nevét!

app/components/card-page/card-page.component.ts:

```
constructor(private cardService: CardService,
            private companyService: CompanyService) { }
ngOnInit() {
    this.getCards();
    this.companyService.getCompanies().subscribe(cmp => this.companies = cmp);
}
companies: Company[];
getCompanyName(id: number) {
    let company = _.find(this.companies || [], c => c.id == id);
    return company && company.name;
}
```

A felületen ezután ki tudjuk kötni a cég nevét az ID alapján:

app/components/card-page/card-page.component.html:

```
<div class="cards-list">
    ...(<i>{{getCompanyName(card.companyId)}}</i>)...
</div>
```

Valamint a részletező nézeten:

```
<h6 class="text-warning"><b>{{selectedCard.role}}</b>
    at <b><a>{{getCompanyName(selectedCard.companyId)}}</a></b></h6>
```

2. FELADAT – MÓDOSÍTÁS

A kártya adatait egyszerűen módosíthatóvá tudjuk tenni. Ehhez módosítsuk a komponenst és a nézetet, és használjuk fel a cégek listázásához az eltárolt cégek tömbjét!

app/components/card-page/card-page.component.ts:

```
selectedCard: Card;
originalCard: Card;
```

```
editing: boolean = false;
cards: Observable<SearchResult<Card>>;

editCard() {
    this.editing = true;
    this.originalCard = this.selectedCard;
    this.selectedCard = { ...this.originalCard }; // make a copy of the object
}
saveCard(card: Card) {
    this.cardService.addOrUpdateCard(card)
        .subscribe(() => {
            this.editing = false;
            this.getCards();
            this.selectedCard = undefined;
            this.originalCard = undefined;
        });
}

cancel() {
    this.editing = false;
    this.selectedCard = this.originalCard;
    this.originalCard = undefined;
}
confirmDelete() {
    if (confirm(`Are you sure you want to delete card for
        ${this.selectedCard.firstName} ${this.selectedCard.lastName}?`)) {
        this.cardService.deleteCard(this.selectedCard.id).subscribe(() => {
            this.editing = false;
            this.getCards();
            this.selectedCard = undefined;
            this.originalCard = undefined;
        });
    }
}
```

app/components/card-page/card-page.component.html:

```
<div class="col-lg-7 panel panel-warning current-card" *ngIf="selectedCard">
    <!-- Here comes the selected or newly created card's editor / detail. -->
    <div class="panel-body">
        <img class="profile-picture pull-right" [src]="selectedCard.imageUrl">
        <input class="form-control"
               *ngIf="editing" [(ngModel)]="selectedCard.imageUrl">
        <h4 class="text-warning">{{selectedCard.firstName}}
            {{selectedCard.lastName}}
            <i class="glyphicon glyphicon-pencil"
               (click)="editCard()" *ngIf="!editing"></i></h4>
        <input class="form-control"
               *ngIf="editing" [(ngModel)]="selectedCard.firstName">
        <input class="form-control"
```

```
                *ngIf="editing" [(ngModel)]="selectedCard.lastName">
        <h6 class="text-warning">"
            <b>{{selectedCard.role}}</b> at
            <b><a>{{selectedCard.companyId}}</a></b></h6>
        <input class="form-control"
                *ngIf="editing" [(ngModel)]="selectedCard.role">
        <select class="form-control" *ngIf="editing"
                [(ngModel)]="selectedCard.companyId">
            <option *ngFor="let company of companies"
                    [ngValue]="company.id">{{company.name}}</option>
        </select>

        <p class="text-info">{{selectedCard.motto}}</p>
        <input class="form-control"
                *ngIf="editing" [(ngModel)]="selectedCard.motto">
        <button class="btn btn-default"
                *ngIf="editing" (click)="saveCard(selectedCard)">Save</button>
        <button class="btn btn-warning"
                *ngIf="editing" (click)="cancel()">Cancel</button>
        <button class="btn btn-danger pull-right"
                *ngIf="editing && selectedCard.id" (click)="confirmDelete()">
                Delete</button>
    </div>
</div>
```

3. FELADAT – LAPOZÁS ÉS SZŰRÉS

A lapozás használatához kezelnünk kell a balra-jobbra lapozást annak függvényében, vannak-e még elemek, valamint az oldalak számának kijelzését kell megvalósítanunk.

app/components/card-page/card-page.component.ts:

```
getCards() {
    this.cards = this.cardService.getCards({ page: this.currentPage });
    this.cards.subscribe(r => {
        this.maxPages = Math.ceil(r.allResults / 5);
        this.currentPage = r.page;
    });
}

currentPage: number = 0;
maxPages: number = 0;
```

app/components/card-page/card-page.component.html:

```
<div class="paging">
    <button class="col-xs-4 btn btn-info"
            [disabled]="currentPage <= 0"
            (click)="currentPage=currentPage-1;getCards()">
        <i class="glyphicon glyphicon-chevron-left"></i>
```

```
    </button>
    <div class="col-xs-4 page-numbers">
        {{currentPage + 1}}/{{maxPages}}
    </div>
    <button class="col-xs-4 btn btn-info"
            [disabled]="currentPage >= maxPages - 1"
            (click)="currentPage=currentPage+1;getCards()">
        <i class="glyphicon glyphicon-chevron-right"></i>
    </button>
</div>
```

A kereséshez szintén nem túl sok dolgot kell elvégeznünk: az input modelljét kell eltárolnunk, az Enter billentyű leütését kezelni, és megfelelően frissíteni a találati listát.

app/components/card-page/card-page.component.ts:

```
searchTerm: string = "";
getCards() {
    this.cards = this.cardService.getCards({
        page: this.currentPage,
        searchTerm: this.searchTerm });
    this.cards.subscribe(r => {
        this.maxPages = Math.ceil(r.allResults / 5);
        this.currentPage = r.page;
    });
}
```

app/components/card-page/card-page.component.html:

```
<input class="form-control"
       placeholder="Filter cards..."
       [(ngModel)]="searchTerm"
       (keyup.enter)="currentPage=0;getCards()">
```

Ezzel az Enter lenyomására megtörténik a keresés (kereshetünk vezeték- és keresztnévre vagy beosztásra), a találatok is ennek megfelelően lesznek kitöltve.

4. FELADAT – ÖNÁLLÓ FELADAT

A tanult megközelítések használatával valósítsd meg az alábbiakat:

1. Készítsd el a kártyák nézetével analóg módon a cégek kezelését is, felhasználva a CompanyService szolgáltatást! Az alábbi funkciókat valósítsd meg: cégek listázása, aktuális cég nevének módosítása, a céghez tartozó kártyák számának megjelenítése, új cég felvétele! A cégeket nem kell tudni szűrni és lapozni, és nem szükséges a módosítást visszavonhatóvá tenni.
2. A kártyákon található cég nevére kattintva navigálj a megfelelő URL-re (**/company/:id**), ahol toltődjön be a kiválasztott cég a részletező nézetbe! Használd a **routerLink** direktívát!