ANGULAR – 4.ČASŤ

Peter Gurský, peter.gursky@upjs.sk

Dorobíme login komponent

- Ak príde **true**, môžeme zobraziť inú stránku

```
constructor(private router: Router, ...) { }

this.router.navigateByUrl('/users');     // v subscribe
```

- Ak príde **false**, zobrazíme hlášku o zlom hesle
- Ak príde chyba, zobrazíme hlášku o chybe komunikácie

Dorobíme si navigáciu

- Vytvoríme komponent obsahujúci navigáciu
  - Nájdeme na Bootstrape príklad na NavBar
- V navigácii dáme linky na všetky zatiaľ používané URL adresy

Životnosť servisu

- Service je singleton
  - Každý komponent, ktorý si ho nechá injektnúť vidí rovnakú inštanciu
- Pri zmene URL cez <a href="">, však dochádza k reštartu celej stránky
  - Service sa vytvára nanovo
  - Token z predchádzajúcej URL už nebude uložený
- Vieme využiť session úložisko v prehliadači
  - Funkcie v globálnom JS objekte **sessionStorage** alebo **localStorage**:
    - setItem(kľúč, hodnota)
    - getItem(kľúč)
    - removeItem(kľúč)
    - clear()
  - max 10MB pre origin (doménu) a iba stringové hodnoty

Router pre Single page application

- Nechceme komunikovať so serverom, keď už aj tak máme natiahnuté všetky komponenty v prehliadači
  - namiesto <a href="/users"> použijeme:
    - <a routerLink="users">
  - router Angularu v tomto prípade iba vymení komponenty, ale nepýta server o novú stránku
  - na presmerovanie v kóde použijeme:

```
import { Router } from '@angular/router';
...
   constructor(private router: Router){}
...
   goToUsers() {
       this.router.navigateByUrl("/users");
       //alebo this.router.navigate(["/users"]); - tu vieme použiť relatívnu cestu
   }
}
```

Nastavovanie class link elementom

- zvýraznenie kliknutého elementu – aktívna linka v menu

```
<a routerLink="/users" routerLinkActive="active">Heroes</a>
```

  - nastavíme class="active"
  - v css nastavíme iný dizajn cez selektor .active

Login / Logout v hlavičke stránky

- Spoločná hlavička: koreňový AppComponent
- Service má komponentu povedať, keď sa stav zmení
  - AppComponent nevie **kedy** to príde a **koľko krát** to príde – je to závislé od toho, čo urobia vnorené komponenty
  - Zaregistrujeme ho na reagovanie na akékoľvek budúce zmeny
  - AppComponent pri každej novej hodnote iba prekreslí linku

Dlhodobé Observable

- Pri vytvorení Observable si zapamätáme jeho Subscriber – objekt ktorý posiela dáta do Observable

```
private loggedUserSubscriber: Subscriber<string>;

 public loggedUser():Observable<string> {
    return new Observable<string>( subcriber => {
      this.loggedUserSubscriber = subcriber;
      subcriber.next(this.user);
    });
  }

 get user() {
    return sessionStorage.getItem("user");
 }
```

```
this.loggedUserSubscriber.next("Jano");
```

ExtendedUsersComponent

- Zapýtame si rozšírených používateľov
  - GET: http://localhost:8080/users/{token}
- Rozšírime triedu User o ďalšie parametre
  - Aj pole objektov typu Group
- Zaevidujeme si komponent v app-routing
- Vypíšeme tabuľku s rozšírenými používateľmi

Refaktor

- Chybové/úspechové hlášky by mohol vypisovať message komponent
  - vypisovať chybové, ale aj pozitívne správy
  - všetky možné chyby komunikácie
    - nedostupný server
    - zlý login alebo heslo
    - nedostatočné práva
  - prebaliť chyby do vlastného message objektu
    - poslať ho message komponentu
      - vypísať správu, nech už bude akákoľvek
    - presmerovať sa na hlavnú stránku?