ANGULAR – 8.ČASŤ

Peter Gurský, peter.gursky@upjs.sk

Nový projekt

- Vytvoríme si nový projekt pre filmy v Angulari 19
  - npm i -g @angular/cli
  - ng new films
- Čo nové nás čaká
  - Angular Material
  - Vlastná knižnica
  - Reaktívne formuláre
  - Vlastné validátory
  - Signály
  - ...

Angular material

- Stránka: https://material.angular.io
- ng add @angular/material
- Každý material komponent je feature modul
- Web prezentácia material komponentu má 3 tabky
  - Overview – prehľad o to, čo je to za komponent
  - API
    - rozpis všetkých súčastí modulu
    - pre nás dôležitá časť: import modulu
  - Examples – príklady použitia aj so zdrojákmi

Login komponent

- Vyrobíme si login komponent v material dizajne
- Potrebujeme komponenty:
  - Card
  - Input
  - Form field
  - Button
  - Icon
- Zabalíme si ich do importno-exportného modulu pre jednoduchšie importovanie aj ďalších našich komponentov
  - V src/modules
    - ng g m material --flat
    - V importe aj exporte uvedieme žiadané moduly material komponentov
    - MaterialModule môžeme importovať v komponentoch
- Na import http klienta treba v app.config.ts dodať `provideHttpClient()`

Konfiguračné premenné

- Vlastnosti, ktoré chceme inak vo vývoji a inak pri nasadení chceme mať v konfiguračnom súbore
- ng g environments
- Pri vývoji sa použije súbor `environment.development.ts`
  - ng serve
- Pri vytváraní balíčka pre server sa použije súbor `environment.ts`
  - ng build
- Pri importovaní v kóde používame cestu k `environment.ts`

environment.development.ts:

```
export const environment = {
  serverUrl: "http://localhost:8080/"
};
```

Angular knižnice

- Vytvoríme samostatný javascriptový balíček
  - Je možné ho poslať do npm úložiska (treba vymyslieť unikátne meno)
  - A následne importovať do ľubovoľného angular projektu cez npm i balíček
- V hlavnom adresári projektu vytvoríme knižnicu
  - ng g library kasv-message
  - Chceme vypisovať pozitívne a negatívne hlášky a spracovávať http chyby
    - Použijeme komponent SnackBar

Čo knižnica obsahuje

- src/public-api.ts
  - Export všetkého, čo chceme nechať používať používateľov našej knižnice
    - Komponenty, servisy, moduly,...
  - To, čo sa neexportuje, zvonku nie je vidieť
- src/lib/*
  - Naše zdrojové súbory, ktoré naprogramujeme
- package.json
  - V peerDependencies uvedieme závislosť na knižniciach
  - Vieme upraviť názov a verziu knižnice
- README.md
  - Informácie, ktoré sa zverejnia ako popis na portáli npmjs.com

Ako knižnicu použiť

- Ak chceme použiť knižnicu v projekte musíme ju vybuildovať
  - ng build kasv2024-message
- Potom vieme normále importovať do modulu alebo komponentu v sekcii imports všetko, čo sme exportovali
- Export do npm úložiska z dist adresára knižnice
  - npm adduser váš_login
  - npm publish

NavBar

- Použijeme Mat-toolbar
- Na informovanie o tom, kto je prihlásený použijeme signál
  - V usersService:
    - `loggedUser = signal(this.username);`
  - V navbar.component.ts:
    - `usersService = inject(UsersServerService);`
    - `userName = this.usersService.loggedUser;`
  - V navbar.component.html:
    - `{{userName()}}`
- Importujeme
  - `MaterialModule, RouterLink, RouterLinkActive`

Writable Signály – základná funkcionalita

- Obal, ktorý umožňuje sledovať zmeny premennej
  - Vytvorenie
    - `const count = signal(0);`
  - Čítanie
    - `console.log('The current count is: ' + count());`
  - Nastavovanie
    - `count.set(2);`
    - `count.update(value => value + 1);`
  - Reagovanie na zmenu
    - `effect(() => console.log('Count = ' + count()));`
  - Závislý signál
    - `const doubleCount = computed(() => count() * 2);`

linkedSignal

- Kombinácia computed signálu a prepisovateľného signálu
  - `const selectedItem = linkedSignal(() => this.items()[0])`
  - Viem spustiť ručnú zmenu, ktorá zmení vypočítanú hodnotu na akú chcem
    - `selectedItem.set(myItem)`
  - Ale ak sa zmení hodnota signálu this.items, znova sa zmení aj selectedItem na prvý item

rxResource a resource (typu ResourceRef)

- Obálka pre signál, ktorý je závislý na asynchrónnom zdroji dát (napr. serveri)
- ```
  const entita = rxResource({
    request: () => myQuerySignal(),
    loader: (myQuery) =>
              this.http.get<Entita>('/entita/' + myQuery)
  });
  ```
- Ak sa udeje zmena v myQuerySignal, spustí sa loader s hodnotou tohto signálu a výsledok prúdu sa po čase uloží v `entita`
- Pozn: loader vezme iba prvú hodnotu
  ```

rxResource a resource (typu ResourceRef)

- vlastnosti, ktoré resource postykuje
  - isLoading – signál (boolean) obsahujúci, či sa práve deje loader
  - value – writable signál obsahujúci aktuálnu hodnotu
  - error – signál obsahujúci chybový objekt, ak cez prúd prišla chyba
  - status – signál so stavom, možné hodnoty
    - ResourceStatus.Idle
    - ResourceStatus.Loading
    - ResourceStatus.Error
    - ResourceStatus.Resolved
    - ResourceStatus.Reloading
    - ResourceStatus.Local, ak hodnota value bola nastavená lokálne