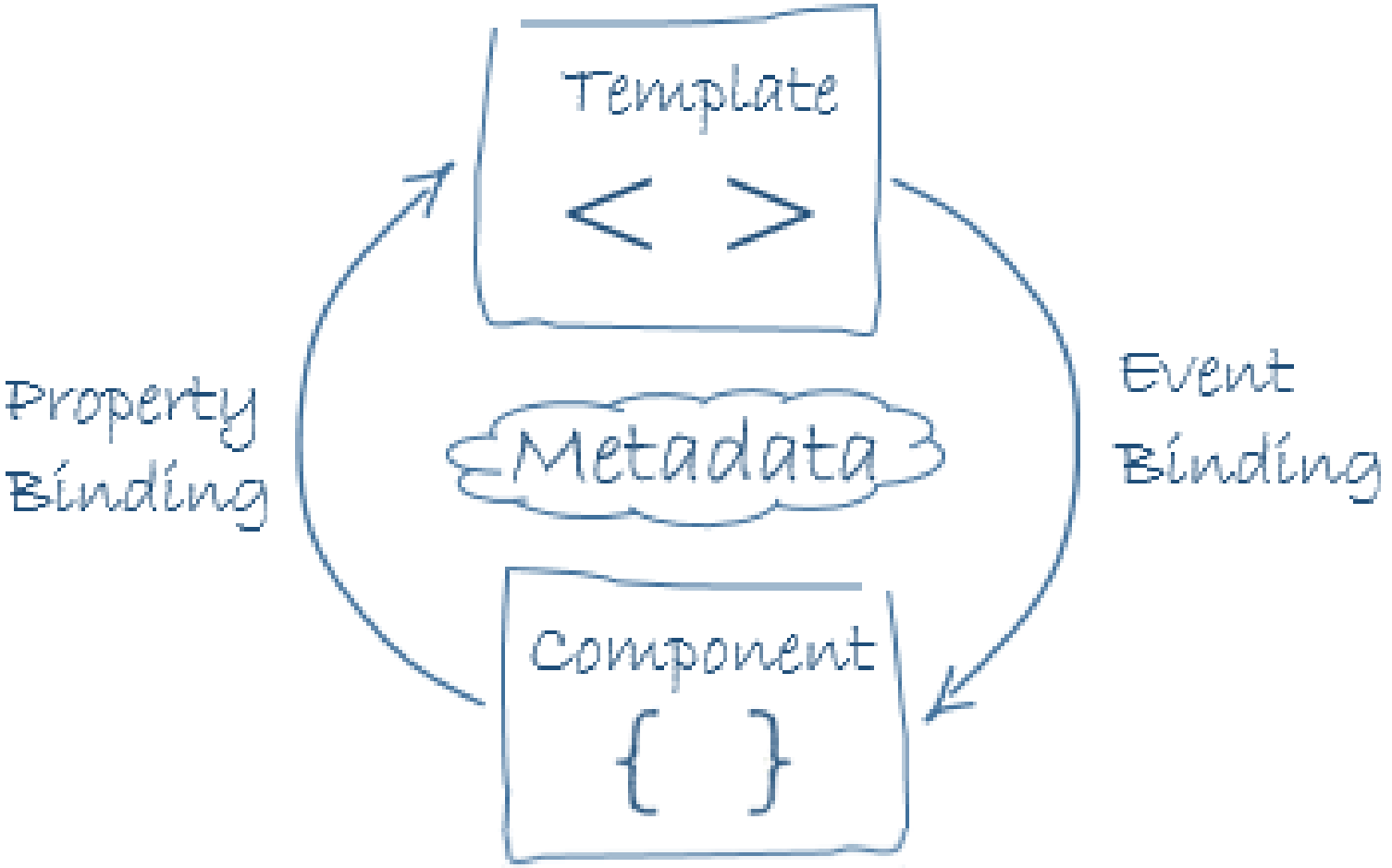
ANGULAR – 2.ČASŤ

Peter Gurský, peter.gursky@upjs.sk

Architektúra komponentov Angularu



Služby (Services)

- Časom môžeme mať viac komponentov, ktoré potrebujú používateľov
- Používateľov nemá spravovať komponent, ale perzistentná vrstva na serveri
- Náš cieľ - vytvoriť službu zodpovednú za zoznam používateľov
  - bude komunikovať s REST serverom
  - a sprostredkovávať tak pre komponenty CRUD operácie nad používateľmi na serveri
- Najprv si spravíme komunikáciu služby s komponentmi
  - Služba bude zatiaľ poskytovať len svoje lokálne pole používateľov

Vytvorenie služby

- Prejdeme do adresára, kde chceme mať službu, napr. src/app/services
- Spustíme príkaz
  - ng g service users
- Vzniknú 2 súbory
  - src/app/services/users.service.spec.ts
  - **src/app/services/users.service.ts**
    - Vytvoríme getter vracajúci pole používateľov

Komponent potrebuje službu - injektneme

app/users/users-component.ts

```
import { UsersService } from '../services/users.service';
...
export class UsersComponent {
 constructor(private usersService: UsersService) {}
}
```

Alternatívne:

```
export class UsersComponent {
usersService = inject(UsersService);
}
```

app/users/users-service.ts

```
@Injectable({
providedIn: 'root'
})
export class UsersService {

}
```

priama registrácia v app.module od Angular 6

Komponent potrebuje service

app/users/users-component.ts

```
export class UsersComponent implements OnInit {

 constructor(private usersService: UsersService) {}

  ngOnInit() {
      this.updateUsers();
  }

  updateUsers() {
      this.users = this.usersService.getUsers();
  }
}
```

Čo môže trvať dlho* nerobíme v konštruktore!

* napr. komunikácia so serverom

Túto metódu musíme v službe vytvoriť

Príprava na dlhé čakanie na dáta

- Synchrónne volanie:
  - Component si vypýta dáta od servisu a čaká
  - Service si vypýta dáta zo servera a čaká
  - Pokiaľ sa čaká, žiaden JavaScript nefunguje (udalosti používateľa – kliky, editácia,...)
- Asynchrónne volanie
  - Component si vypýta dáta od servisu a zaeviduje funkciu, ktorá sa má spustiť, keď dáta dôjdu
  - Service si vypýta dáta zo servera a zaeviduje funkciu, ktorá sa má spustiť, keď dáta dôjdu
  - Pokiaľ neprídu dáta, všetko funguje

Observable

- Trieda predstavujúca prúd/producenta dát, ktoré prídu v budúcnosti
- Vieme zaregistrovať metódu, ktorá sa má spustiť, keď dáta prídu - cez volanie subscribe(metóda)
- Metóda sa zvykne vkladať ako šípková funkcia
- Moderný spôsob robenia asynchrónnych volaní
- Zmeňme metódu v službe:
  - Operátor „of“ vyrobí nový objekt typu Observable
  - Musíme ho importovať

```
import { Observable, of } from 'rxjs';

getUsers(): Observable<User[]> {
    return of(this.users);
}
```

Vyrobíme prúd dát, ktoré sa pošlú, keď na konci niekto zavolá subscribe. Tieto dáta dodáme cez parameter of().

Zaregistrovanie poslucháča v komponente

app/users/users-service.ts

```
getUsers(): Observable<User[]> {
      return of(this.users);
}
```

app/users/users-component.ts

```
updateUsers() {
   this.usersService.getUsers().subscribe(users => this.users = users);
}
```

Keď sa dáta z Observable prídu, spustí sa naša funkcia. Dáta dostane ako vstupný parameter users.

REST server

- Použijeme predpripravný REST server,
  - stiahneme si súbor films-server.jar
    - java -jar films-server.jar
- Keď ho spustím počúva na adrese http://localhost:8080/
  - Vyskúšame cez Postman-a alebo Thunder client-a http://localhost:8080/users
- Naša Angular aplikácia je prístupná cez NodeJS server na http://localhost:4200/

Pripravíme si Angular

app.config.ts

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

export const appConfig: ApplicationConfig = {
  providers: [provideRouter(routes),
              provideHttpClient()
             ]
}
```

getUsers() cez AJAX volanie

app/users/users-service.ts

```
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs/Observable';
...
constructor(private http: HttpClient) { }

private restServerUrl: string = "http://localhost:8080/users";

getUsers(): Observable<User[]> {
    return this.http.get<User[]>(this.restServerUrl);
}
```

Injektneme inštanciu HttpClient (alebo použijeme inject)

Hlúpe pretypovanie objektu ktorý vznikol z JSONu.
Typescript sme presvedčili, že objekt bude mať rovnakú štruktúru, ako naša trieda User, ale nikde sa to neoveruje.

Ak chceme skutočné objekty triedy User, musíme ich vyrobiť

app/users/users-service.ts

```
import { map } from 'rxjs/operators';
...
getUsers(): Observable<User[]> {
    return this.http.get<User[]>(this.restServerUrl + 'users')
        .pipe(map(response => this.handleGetUsersResponse(response)));
}

private handleGetUsersResponse(jsonUsers:User[]):User[] {
    let remoteUsers:User[] = [];
    for (let jsonUser of jsonUsers) {
        remoteUsers.push(User.clone(jsonUser));
    }
    return remoteUsers;
}
```

Ak chceme skutočné objekty triedy User, musíme ich vyrobiť

app/users/users-service.ts

```
import { map } from 'rxjs
...
getUsers(): Observable<
    return this.http.get<Us
        .pipe(map(respon
}

private handleGetUsersResponse(jsonUsers:User[]):User[] {
    let remoteUsers:User[] = [];
    for (let jsonUser of jsonUsers) {
        remoteUsers.push(User.clone(jsonUser));
    }
    return remoteUsers;
}
```

```
export class User {
  public static clone(u: User): User {
    return new User(u.name, u.email, u.id,
                    u.lastLogin, u.password);
  }
  ...
}
```

Chyby pri Observable

- Ak sa komunikácia nepodarí, funkcia vložená cez subscribe sa nespustí, ale vyletí chyba do konzoly
- Ak chceme túto chybu odchytiť, môžeme v subscribe vložiť **objekt** s dvoma záchytnými funkciami, druhá funkcia je na spracovanie chyby, ktorú dostane na svojom vstupe

app/users/users-component.ts

```
updateUsers() {
  this.usersService.getUsers().subscribe({
    next: users => this.users = users,
    error: error => console.log('chyba komunikácie: ' + JSON.stringify(error))
  });
}
```

Zobrazme spätnú väzbu používateľovi

- Na zobrazenie môžeme použiť Bootstrap triedu alert
- Zobrazíme takýto alert (pozitívny alebo negatívny) ako súčasť stránky,
- Keď Observable dodá dáta alebo chybu - zobrazíme zmysluplnú hlášku používateľovi