Angular Identity Management

Le service Users

Un service est une classe “simple” accessible dans toute l’application grâce à l’injection de dépendance.
Cette classe manipule des données (API) tels que définis par un service Web REST avec la classe HttpClient

Plutôt que d’utiliser de “fausses” données directement, nous allons créer un service pour obtenir les utilisateurs LDAP avec le fichier “ldap-mock-data.ts”.

Création du service

Nous allons d’abord créer un dossier pour le service puis générer le service

- Créez le dossier services à la racine du projet, commande : `mkdir src/app/service`
- Placez-vous dans le dossier
- Créez le service UsersService avec la commande suivante : `ng generate service users`

Explications pour le fichier “users.service.ts”:

- **@Injectable**({ providedIn: 'root' }) : Le décorateur Injectable indique que cette classe doit être “répertoriée” par le mécanisme Dependency Injection de Angular.
    - providedIn: 'root' : indique que le service est disponible à la racine de l’injecteur principal (donc du projet)

Nous allons ajouter la liste des utilisateurs en attribut statique pour “simuler“ un Web Service (ou API).

```
export class UsersService {
  // Liste des utilisateurs
  users: UserLdap[] = LDAP_USERS;
  ...
```

Obtenir tous les utilisateurs

Dans la classe, créez une méthode nommée “getUsers”, le code sera le suivant :

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

N’oubliez pas d’ajouter les imports !

Explications:

- Observable: Retourne une liste à laquelle une classe pourra s’abonner
    - UserLdap[] : type de valeurs retournées
    - of(UsersService.users) : transforme le tableau users en liste observable

Obtenir un utilisateur

Toujours dans la même classe, ajoutez le code suivant :

```
getUser(login: string): Observable<UserLdap | undefined> {
  return of (this.users.find(user :UserLdap  => user.login === login));
}
```

Modification du composant ldap-list

Le composant LdalListComponent ne doit plus utiliser directement la liste mais doit utiliser le service. Nous allons modifier le fichier “ldap-list.component.ts” comme ceci:

- Supprimez la ligne import {*LDAP_USERS*} from "../model/ldap-mock-data";
- Modifiez le constructeur pour créer le service avec l’Injection de Dépendances :

```
constructor(private usersService: UsersService, private router: Router)
{
}
```

- Modifiez la méthode getUsers :

```
private getUsers(): void {
  this.usersService.getUsers().subscribe(
      users => {
        if (this.unactiveSelected) {
          this.dataSource.data = users.filter( user =>
              user.active === false
          );
        } else {
          this.dataSource.data = users
        }
      });
}
```

La méthode getUsers() du service UsersService retourne un Observable. Afin d’obtenir les données, il faut utiliser la méthode subscribe qui sera appelée lorsque les données seront disponibles.

Liens

https://angular.io/start

https://blog.angular.io/

https://blog.angular-university.io/

https://guide-angular.wishtack.io/

https://openclassrooms.com/fr/courses/4668271-developpez-des-applications-web-avec-angular

https://www.typescriptlang.org/docs/home.html

https://blog.soat.fr/

Parent Child Two way binding
https://medium.com/@preethi.s/angular-custom-two-way-data-binding-3e618309d6c7

Organisation par module
Mise en place de la sécurité
https://angular.io/guide/route

Login:
https://loiane.com/2017/08/angular-hide-navbar-login-page/

Dialog:
https://blog.angular-university.io/angular-material-dialog/

Angular Material
https://material.angular.io/

https://medium.com/@ismapro/first-steps-with-angular-7-with-angular-cli-and-angular-material-d69f55d8ac51

https://www.positronx.io/create-angular-material-8-custom-theme/

https://akveo.github.io/ngx-admin/

https://auth0.com/blog/creating-beautiful-apps-with-angular-material/