Angular Identity Management

Edition d'un élément LdapUser

Nous allons modifier le composant Ldap pour éditer un utilisateur.
Pour ce faire, nous allons :

- créer un nouveau composant pour l'édition
- ajouter la route pour éditer un utilisateur
- modifier le composant pour aller sur la page d'édition en cliquant sur le bouton "Editer" de l'utilisateur

Créer un nouveau composant

Nous allons créer le composant LdapDetailComponent avec la commande suivante : `ng generate component ldap-detail --module=app`. Nous sommes obligé d'utiliser l'option module car nous avons 3 module dans le répertoire src: le module principal, le module pour le routage et le module pour Material. Nous allons écrire le code plus loin.

Ajout de la route

La route que nous allons ajouter est la suivante : http://localhost:4200/users/test.v8
Pour ce faire :
Editer le fichier app-routing.module.ts
Ajouter la ligne suivante (en gras):

```
const routes: Routes = [
  { path: 'users/list', component: LdapListComponent},
  { path: 'user/:id', component: LdapDetailComponent},
  { path: '**', component: PageNotFoundComponent }
];
```

N'oubliez pas de faire l'importation du composant.

Modification du composant LdapDetailComponent

Nous allons ajouter un méthode sur le "click" pour chaque ligne afin d'éditer un utilisateur. Modifier la fin du fichier "ldap-list.component.html" ainsi :

```
    <tr mat-header-row *matHeaderRowDef="displayedColumns"></tr>
    <tr mat-row *matRowDef="let row; columns: displayedColumns;"
        (click)="edit(row.login)"
    ></tr>
</table>
```

Il faut ajouter la méthode edit(login: string) au composant avec le code suivant:

```
  edit(login: string) {
    this.router.navigate(['/user', login]).then( (e) => {
      if (! e) {
        console.log("Navigation has failed!");
      }
    });
  }

}
```

Il faut aussi modifier le fichier app-material.module.ts pour ajouter les ToolTip :

```
@NgModule({
 exports: [
   ...
  MatTooltipModule,
 ]
})
export class AppMaterialModule { }
```

A ce stade vous devriez pouvoir naviguer jusqu'au détail d'un utilisateur (composant LdapDetailComposant) et voir le message "ldap-detail works!".

Obtention du paramètre de l'URL

Pour rappel, la route est de la forme : http://localhost:4200/users/test.v8

Nous allons

- obtenir la route qui a chargé le composant
- obtenir le paramètre (login) de la route

Obtenir le composant de routage

Il faut modifier le constructeur du composant "LdapDetailComponent" comme ceci:

```
constructor(private route: ActivatedRoute) { }
```

Ajouter manuellement, l'importation de la classe Location :

```
import { Location } from '@angular/common';
```

Grâce à l'injection, les objets seront automatiquement créés.

Obtenir le paramètre

Ajouter la méthode getUser appelé depuis ngOnInit pour obtenir le login:

```
ngOnInit(): void {
  this.getUser();

}
private getUser(): void {
  const login = this.route.snapshot.paramMap.get('id');

  console.log("getUser= " + login)
}
```

Obtention de l'utilisateur

Nous allons obtenir l'utilisateur à partir du service UsersService.
Nous allons donc faire les modifications suivantes (toujours dans LdapDetailComponent):

- Ajouter l'attribut suivant:
  user: UserLdap;
- Constructeur:
  ```
  constructor(
    private usersService: UsersService,
    private route: ActivatedRoute,
  ) { }
  ```
- Méthode getUser :
  ```
  private getUser(): void {
    const login = this.route.snapshot.paramMap.get('id');

    this.usersService.getUser(login).subscribe(
        user => { this.user = user; console.log("LdapDetail getUser ="); console.log(user); }
    );
  }
  ```

Édition avec un formulaire

Nous utiliserons les Reactive forms d'Angular.

Dans la classe, nous utiliserons :
- FormBuilder pour construire le formulaire
- FormGroup pour regrouper des contrôles
- FormControl pour créer un contôle

Dans le template HTML, nous utiliserons :
- (optionnel) `[formGroup]='nomDuGroupe"` pour nommer le groupe utilisé dans le formulaire
- La balise `formControlName` pour nommer le contrôle

Voir aussi :
Introduction to forms in Angular
Reactive forms

Le composant

Nous utilisons des composants Material, il faut donc les ajouter à app-material.module.ts (si ce n'est pas déjà fait) :

```
@NgModule({
 exports: [
   ...,
   MatProgressSpinnerModule,
   MatSelectModule,
 ]
})
export class AppMaterialModule { }
```

Nous avons utilisé des évènements et des attributs dans le html, il faut donc les implémenter dans le composant LdapDetailComponent :

```
export class LdapDetailComponent implements OnInit {
  user: UserLdap;
  processLoadRunning = false;
  processValidateRunning = false;

  constructor(
    private usersService: UsersService,
    private route: ActivatedRoute,
    private fb: FormBuilder,
    private router: Router,
  ) { }
```

```
  ngOnInit(): void {
    this.getUser();

  }
  private getUser(): void {
    const login = this.route.snapshot.paramMap.get('id');

    console.log("getUser= " + login)
  }
  private formGetValue(name: string): any { return null; }
  goToLdap() : void {
    this.router.navigate(['/users/list']);
  }

  onSubmitForm(): void  {}
  updateLogin() : void {}
  updateMail(): void  {}
  isFormValid(): boolean  { return false; }
}
```

Reactive Forms

Nous allons utiliser les "Reactive forms" pour gérer les formulaires (obtention des valeurs, validations, …).

Pour cela il faut importer les modules (app.module.ts) :

```
@NgModule({
…
imports: [
 BrowserModule,
 FormsModule,
 ReactiveFormsModule,
 AppRoutingModule,
 BrowserAnimationsModule,
 AppMaterialModule,
],
…
})
export class AppModule { }
```

Dans le composant “ldap-detail.component.ts”, nous allons utiliser une “aide » : le FormBuilder.

```
export class LdapDetailComponent implements OnInit {
  ...
  user: UserLdap;
  processLoadRunning = false;
  processValidateRunning = false;

  userForm = this.fb.group({
    login: [''], // Valeur de départ vide
    nom: [''],
    prenom: [''],
    // Groupe de données imbriqué
    passwordGroup: this.fb.group({
      password: [''],
      confirmPassword: ['']
    }),
    mail: {value: '', disabled: true},
  });

  constructor(
    private usersService: UsersService,
    private route: ActivatedRoute,
    private fb: FormBuilder,
    private router: Router,
  ) { }
```

Pour créer un formulaire avec FormBuilder, il faut :

- Créer l’objet dans le constructeur grâce à l’Injection de Dépendances
- Créer un groupe de données
- (optionnel) Créer un groupe de données imbriqué

Nous allons compléter les méthodes précédentes.

Méthode updateLogin : elle permet de mettre à jour le login lorsque l’utilisateur modifie le nom ou le prénom
Le code :

```
  updateLogin(): void {
    this.userForm.get('login').setValue((this.formGetValue( name: 'prenom')
        + '.' + this.formGetValue( name: 'nom')).toLowerCase());
    this.updateMail();
  }
```

Méthode updateMail : elle permet de mettre à jour le mail lorsque le login est modifié
Le code :

```
updateMail(): void {
  this.userForm.get('mail').setValue(this.formGetValue( name: 'login').toLowerCase()
      + '@epsi.lan');
}
```

Méthode formGetValue :

```
private formGetValue(name: string): any {
  return this.userForm.get(name).value;
}
```

Méthode onSubmitForm :

```
onSubmitForm() {
  // Validation des données (à voir plus tard)
}
```

Modification du HTML

Nous allons utiliser les formulaires et les composants Material dans “ldap-detail.component.html” :

```
<h3 class="mat-h3">
  <button mat-icon-button (click)="goToLdap()">
    <mat-icon aria-label="Accueil">arrow_back</mat-icon>
  </button>
  Edition d'un utilisateur
</h3>

<section class="loading" *ngIf="processLoadRunning==true">
  <mat-spinner diameter="25" ></mat-spinner>
  <span>  En cours de chargement ...</span>
</section>


<form class="user-form" *ngIf="processLoadRunning==false"
      [formGroup]="userForm"
      (ngSubmit)="onSubmitForm()"
>
```

```
<table class="user-full-width">
  <tr>
    <td>
      <mat-form-field class="user-full-width">
        <input matInput class="form-control" placeholder="Nom de l'utilisateur"
               id="nom" formControlName="nom" required
               (input)="updateLogin()"
        />
      </mat-form-field>
    </td>
    <td>
      <mat-form-field class="user-full-width">
        <input matInput class="form-control" placeholder="Prénom de l'utilisateur"
               id="prenom" formControlName="prenom" required
               (input)="updateLogin()"
        />
      </mat-form-field>
    </td>
  </tr>
</table>

<table class="user-full-width">
  <tr>
    <td>
      <mat-form-field class="user-full-width">
        <input matInput class="form-control" placeholder="Login de l'utilisateur"
               id="login" formControlName="login" required
               (input)="updateMail()"
        />
      </mat-form-field>
    </td>
    <td>
      <mat-form-field class="user-full-width">
        <mat-label>Mail</mat-label>
        <input matInput name="mail"
               formControlName="mail"
        />
      </mat-form-field>
    </td>
  </tr>
</table>
```

```
<table class="user-full-width" formGroupName="passwordGroup">
  <tr>
    <td>
      <mat-form-field class="user-full-width">
        <input type="password" matInput
               placeholder="Mot de passe"
               id='password' formControlName='password'
        >
      </mat-form-field>
    </td>
    <td>
      <mat-form-field class="user-full-width">
        <input type="password" matInput class="form-control"
               placeholder="Vérification du mot de passe"
               id="confirmPassword" formControlName="confirmPassword"
        />
      </mat-form-field>
    </td>
  </tr>
</table>

<div style="display:flex; justify-content:flex-end;">
  <button type="submit" mat-stroked-button color="primary"
          [disabled]="processValidateRunning || !isFormValid()" >
    <mat-icon *ngIf="processValidateRunning">
      <mat-spinner diameter="16"></mat-spinner>
    </mat-icon>
    Valider
  </button>
</div>
</form>
```

Le CSS associé :

```css
.loading {
  display: flex;
  align-content: center;
  align-items: center;
}

.user-form {
  min-width: 150px;
  max-width: 750px;
  width: 100%;
}

.user-full-width {
  width: 100%;
  border-collapse: collapse;
}

td {
  padding-left: 12px;
}
```

Testez !! 😊

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/