POLYTECH MARSEILLE Aix*Marseille Université **Informatique – 4A**

Applications web et mobiles

TP n°8 : Ajout de topics

Étape 39 – Le bouton d'ajout de sujet

Vers le haut de la page des *topics* se trouve un bouton « nouveau sujet ». Quand on clique dessus, on souhaite que cela ouvre une fenêtre permettant de saisir ce nouveau sujet. Rajoutez ce bouton dans le fichier `topics.component.html`. Pour cela, utilisez les boutons d'*Angular Material*, cf. l'URL https://material.angular.io/components/button/overview. Reliez l'événement `click` à une méthode `openDialog()` qui, pour l'instant, ne fait rien.

Étape 40 – La fenêtre d'ajout de sujet

Quand on clique sur le bouton, on souhaite maintenant qu'un popup s'ouvre, nous demandant de saisir le sujet. Pour cela, créez un nouveau composant `create-topic-dialog`. L'URL https://material.angular.io/components/dialog/overview vous montre comment créer de tels popups.

⚠ Ce qu'il faut retenir de cet exemple :

- Dans le HTML `topics.component.html` :
  - l'étape 39 a déjà tout mis en place.
- Dans le TypeScript `topics.component.ts` :
  - la classe `DialogOverviewExample` correspond à votre `TopicsComponent` ;
  - la classe `DialogOverviewExampleDialog` correspond à votre `CreateTopicDialogComponent` ;
  - le champ `data` passé en argument de `this.dialog.open` contient toutes les données que vous souhaitez transmettre au popup (ici, seul l'ID du cours est utile).
- Dans le HTML `create-topic-dialog.component.html` :
  - il est inutile, ici, d'utiliser les `<mat-form-field>` ;
  - remplacez le `[mat-dialog-close]` par le traditionnel événement `click`.
- Dans le TypeScript `create-topic-dialog.component.ts` :
  - il faut écrire les callbacks des événements `click` des deux boutons (Fermer/Créer le sujet).

Vous pouvez donc adapter la méthode `openDialog` de l'exemple pour vos besoins et écrire le contenu du composant `create-topic-dialog`. Pour l'instant, quand on clique sur les boutons, on se contente d'exécuter `this.dialogRef.close();` afin de fermer le popup.

Étape 41 – Création du nouveau sujet

Dans la *callback* du bouton « Créer le sujet », si le sujet saisi est non vide, faites en sorte de transmettre le nouveau sujet en question à votre *backend*. Si votre *backend* vous a transmis un message d'erreur, affichez-le au dessus des boutons « Fermer » et « Créer le sujet », sinon affichez sa réponse dans la console et fermez la fenêtre.

Sur l'URL `https://material.angular.io/components/dialog/overview`, juste après le premier exemple, regardez l'exemple de `dialogRef.afterClosed()`, qui montre comment récupérer des données du popup transmises via `dialogRef.close(données)`. Faites en sorte que le `TopicsComponent` affiche ces données dans la console.

Étape 42 – Prise en compte du nouveau sujet dans la liste des topics

Dans le composant `TopicsComponent`, rajoutez le nouveau sujet à votre table de topics. Pour cela, le plus simple est de procéder de la manière suivante :

1. dans la méthode `ngOnInit`, ne pas affecter directement ce qui a été renvoyé par `sendMessage` à `this.dataSource.data`, mais plutôt stocker cette information dans un attribut de la classe `TopicsComponent`. Appelons `topics` cet attribut ;
2. toujours dans `ngOnInit`, après avoir initialisé `this.topics`, affectez le à `this.dataSource.data` ;
3. dans le `dialogRef.afterClosed().subscribe` de la méthode `openDialog`, ajoutez le nouveau sujet au tableau `this.topics` via la méthode `push` ou `unshift`. Puis réaffectez `this.topics` à `this.dataSource.data`.

Optionnel : pour ceux/celles qui ont fini en avance

La page des *posts* est similaire à celle des *topics*. Sa création revient essentiellement à faire du copier/coller des composants des *topics*. La seule différence majeure est l'utilisation d'un éditeur de texte pour saisir les *posts*, ce qui correspond à la dernière étape de ce TP.

Étape 43 – Backend : récupération de la liste des posts par PHP

Dans votre *backend*, écrivez un script PHP qui vous renvoie la liste des posts associés à un topic.

Étape 44 – Le code TypeScript

Mettez à jour votre classe `PostsComponent`. Cela revient essentiellement à faire du copier-coller de ce que vous avez écrit dans la classe `TopicsComponent`.

Étape 45 – Template HTML

Copiez-collez le template HTML de `topics.component.html`, notamment la partie relative au breadcrumb et au bouton d'ajout et adaptez ce code (mais, pour l'instant, dans le bouton, conservez l'`<input>` pour saisir le post, vous changerez cela plus tard). Pour cela, il faudra également que vous copiez-colliez et que vous adaptiez le code du `create-topic-dialog`.

Étape 46 – L'éditeur de posts

L'URL https://ckeditor.com/docs/ckeditor5/latest/builds/guides/integration/frameworks/angular.html décrit comment inclure l'éditeur *rich text* « *ckeditor* » dans *Angular*. Essentiellement, cela revient à réaliser les opérations suivantes :

- dans le fichier `create-post-dialog.component.ts` :
  - ajouter « `CKEditorModule` » dans la liste des imports ;
  - ajouter « `import ClassicEditor from '@ckeditor/ckeditor5-build-classic';` » au début du fichier ;
  - ajouter l'attribut « `public Editor = ClassicEditor;` » dans la classe `CreatePostsDialogComponent`.
- dans le fichier `create-post-dialog.component.html` :
  - remplacer l'`<input>` de la fenêtre modale par :
    ```
    <ckeditor [editor]="Editor" [(ngModel)]="newPost"></ckeditor>
    ```
- dans le fichier `create-posts-dialog.scss` :
  - ajouter les lignes suivantes si vous souhaitez augmenter la hauteur de votre éditeur de posts :
    ```
    ::ng-deep .ck-editor__editable_inline {
        min-height: 20ex !important;
        max-height: 20ex !important;
    }
    ```

⚠ Comme vous transpilez *Angular* en mode strict, il faut rajouter un fichier `typings.d.ts` dans le répertoire `src/app` avec le contenu suivant, qui indiquera à TypeScript le type de votre `ClassicEditor` :

```
declare module '@ckeditor/ckeditor5-build-classic' {
  const ClassicEditorBuild: any;
  export default ClassicEditorBuild;
}
```