POLYTECH MARSEILLE
Aix*Marseille Université

Informatique – 4A

Applications web et mobiles

TP n°5 : Page de cours

Étape 26 – La table des cours

Dans votre composant `CoursComponent`, vous avez récupéré la liste des cours. Maintenant, il convient de l'afficher. Pour cela, le plus simple est d'utiliser une table d'*Angular Material*, cf. l'URL `https://material.angular.io/components/table/overview`. Le premier exemple de cette page vous montre comment construire une telle table. Notez qu'en cliquant en haut à droite de l'exemple sur l'icône `<>`, vous pourrez visualiser le template HTML et le code TypeScript de l'exemple.

Le template HTML est simple à comprendre : les balises `<ng-container></ng-container>` définissent ce que vont contenir les colonnes de votre table. Les `matColumnDef="..."` donnent un identifiant à chaque colonne. Les `mat-header-cell` indiquent clairement le titre de la colonne. Les `<td mat-cell *matCellDef="let element"> {{element.xxxx}} </td>` sont un peu plus compliqués à comprendre : si vous observez bien, les attributs `xxxx` de l'exemple sont, respectivement, `position`, `name`, `weight` et `symbol`. Cliquez sur l'onglet TS afin de visualiser le code TypeScript. Vous pouvez alors voir que ces champs sont ceux de l'interface `PeriodicElement`. En dessous de cette interface, vous avez le tableau Javascript des données qui seront affichées dans la table. Chacune des lignes de ce tableau Javascript correspond à un objet vérifiant cette interface. Les balises `<td mat-cell *matCellDef="let element"> {{element.xxxx}} </td>` indiquent donc que, si l'on veut afficher un `element` de type `PeriodicElement` dans une colonne donnée, il faudra afficher dans cette colonne la valeur de l'attribut `xxxx` de cet `element`.

À la fin du template, la ligne `<tr mat-header-row *matHeaderRowDef="displayedColumns"></tr>` précise les colonnes qui vont être attendues par *Angular* lorsqu'il va remplir chaque ligne de données. Si vous regardez le TypeScript, vous verrez que `displayedColumns` est un attribut du composant `TableBasicExample` qui recense les identifiants des colonnes spécifiés dans les `matColumnDef` des `<ng-container></ng-container>`. La dernière ligne du template HTML (`<tr mat-row ...>`) précise qu'il faut afficher tout le contenu de la table JavaScript : en fait, le `let row` correspond, en l'essence, à un `*ngFor="let row of ELEMENT_DATA"`.

Enfin, dans le TypeScript, notez que le tableau `ELEMENT_DATA` est copié dans un attribut `datasource`, qui est précisément ce que l'on passe en attribut à la balise `<table>` sur la première ligne du template HTML. Pour tester, créez un tableau `ELEMENT_DATA` contenant un seul cours et initialisez l'attribut `datasource` avec, comme indiqué sur la page web d'*Angular Material*.

Chaque fois que vous utilisez des composants d'*Angular Material*, il faut importer des modules dans `app.module.ts`. Pour déterminer ce qu'il faut importer, cliquez, tout en haut de la page d'*Angular Material* sur l'onglet API. Vous verrez alors l'import à faire.

Voilà, maintenant, vous savez comment utiliser les tables d'*Angular Material*. Si vous utilisez des

composants d'*Angular Material*, je vous suggère de rajouter un thème d'*Angular Material* dans la rubrique « styles » du fichier `angular.json` à la racine de votre application. Personnellement, j'ai rajouté :

```
"./node_modules/@angular/material/prebuilt-themes/indigo-pink.css"
```

Mais d'autres couleurs plus chatoyantes peuvent être utilisées.

Créez donc la table *Angular Material* dont vous avez besoin pour afficher la liste des cours suivis par l'utilisateur.

Étape 27 – Remplissage de la table des cours

Supprimez `ELEMENT_DATA`. Vous allez maintenant remplir comme il faut l'attribut `datasource` avec la liste des cours que vous avez reçue de votre *backend* dans l'étape 25. Pour cela, il faut créer correctement le `datasource`. À cet effet, reportez-vous à l'exemple de la section *Pagination* de la page d'*Angular Material* que vous avez lue à l'étape précédente. Vous pouvez voir dans le TypeScript de l'exemple que `datasource` est une instance de la classe `MatTableDataSource`. Il faut donc la créer comme tel. Cette classe a un attribut `data` qui contient les données du datasource. Vous pouvez modifier sa valeur (`this.datasource.data = ...`) afin de remplir dans la table de votre application de nouvelles données. Faites donc cette opération fin de placer dans votre `datasource` tous les cours qui vous ont été retournés par votre *backend*.

Étape 28 – Pagination

Dans le forum de démonstration, vous pouvez observer qu'en bas des tables, il y a un composant permettant de découper la table en plusieurs pages (un *paginateur*). La section `pagination` de l'étape précédente vous montre comment inclure ce composant. Le TypeScript de l'exemple de cette section montre qu'il faut légèrement modifier votre classe `CoursComponent`. Tout d'abord, il vous faudra importer `MatPaginator`. Ensuite, vous pouvez voir une ligne :

```
@ViewChild(MatPaginator) paginator!: MatPaginator;
```

Le décorateur `@ViewChild` indique que le composant `CoursComponent` a un enfant de type `MatPaginator` que vous appellerez `paginator` dans votre classe `CoursComponent`. Cet enfant correspond à la balise `<mat-paginator ...></mat-paginator>` du template HTML. Quand vous regardez le forum de démonstration, le paginateur indique combien votre table contient de cours. Pour que `paginator` ait cette information, il faut qu'il soit relié à votre `datasource`. C'est le but de la ligne :

```
this.dataSource.paginator = this.paginator;
```

Le `this.paginator` correspond à l'objet que vous avez référencé via le `@ViewChild`. L'idée, dans votre cas, n'est pas d'écrire cette ligne dans une méthode `ngAfterViewInit`, mais plutôt dans le `ngOnInit`, juste après la ligne où vous avez mis à jour l'attribut `data` du `datasource`. Normalement, maintenant, vous devez voir votre paginateur indiquer le nombre de cours que l'utilisateur suit.

Étape 29 – Tri des colonnes

La section suivante de la documentation d'*Angular Material* (*Sorting*) concerne la possibilité de trier les colonnes de vos tables *Angular Material*. Ici, il n'y a pas grand chose à faire : il faut juste rajouter dans le template HTML un attribut `matSort` à la balise `<table>` et un attribut `mat-sort-header` dans la description de chacune des colonnes.

Côté TypeScript, il faut rajouter un `@viewChild` pour accéder au composant de tri, relier ce composant au `datasource` (dans la méthode `ngOnInit`, comme dans l'étape précédente, plutôt que dans la méthode `ngAfterViewInit`). Enfin, rajouter l'import du `MatSortModule` dans `app.module.ts`. Comme vous pouvez le constater, avec les composants d'*Angular Material*, il faut souvent, mais pas tout le temps, rajouter des imports dans `app.module.ts` (par exemple, on ne rajoute pas de module pour les `MatTableDataSource`). Pour déterminer quand on doit le faire, c'est souvent simple : si, dans la documentation, vous voyez un lien vers l'*API doc*, c'est qu'il faudra faire un import. À la fin du 1er paragraphe de la section *Sorting*, vous pouvez observer ce lien.

Vérifiez que, dans les titres des colonnes de votre table, vous pouvez bien retrier les éléments du tableau.