POLYTECH
NICE SOPHIA

TD Techno Web

Démarrage et environnement

Dans ce TD vous allez devoir manipuler tous les concepts de base dont vous aurez besoin lors de votre projet.

Pour commencer il faut installer NodeJS sur votre machine si ce n'est pas déjà fait: https://nodejs.org/en/download/

Côté frontend

Clonez le repository

```
git clone https://github.com/NablaT/AngularStarter
```

Installez les dépendances listées dans le *<package.json>* nécessaires au projet:

```
npm install
```

Lancez la compilation et le serveur

```
npm start
```

Step 1 : Création d'une page

Le but est de mettre à jour le starter pour commencer votre application pour la signalisation et la visualisation d'événements.

Au lieu de récupérer une liste de messages, vous devez afficher une liste d'événements et pouvoir cliquer sur l'un d'entre eux pour voir plus d'informations.

Step 2 : Créer un service pour gérer les interactions serveur

Créer un service afin de gérer les interactions faites avec la partie serveur. Ce service va donc se charger d'effectuer les calls HTTP. Un exemple de service contenant un call HTTP est déjà

présent dans le starter. Il peut vous servir de base pour créer votre service permettant de récupérer la liste d'évènements.
Plus tard, c'est dans ce service que vous pourrez ajouter ce qu'il faut pour éditer/modifier/supprimer un évènement. *(Il faudra avoir fait les étapes "Côté serveur" d'abord)*

Step 3 : Boucle sur la liste d'évènements

Une fois que vous arrivez à récupérer la liste d'évènements. Vous allez pouvoir l'afficher dans votre page. Pour se faire nous avons besoin de faire une boucle sur la list d'évènements. Les boucles en Angular se font avec la directive native ngFor.
(https://angular.io/api/common/NgForOf)

Voici un exemple vu dans le cours:

```
<div *ngFor="let event of eventList">
{{event}}
</div>
```

Step 4 : Communication Component-Component

Un element Event est ici lié à la liste d'événement. L'affichage d'un seul événement n'est donc pas réutilisable en dehors de la liste.

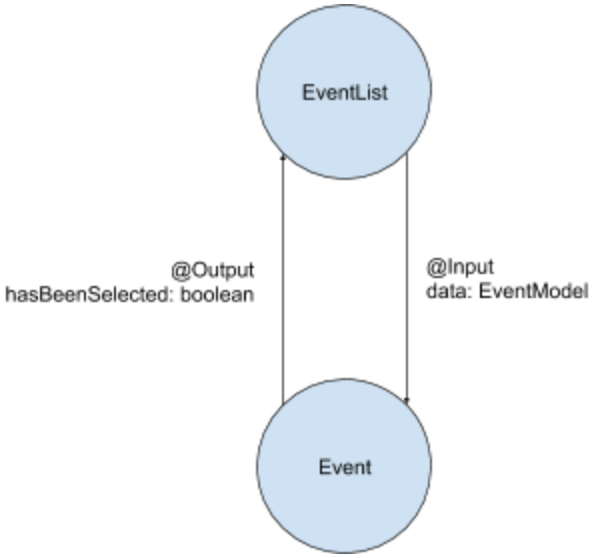
Le but est donc de séparer l'événement de sa liste avec deux composants:

- Composant Père: Event liste: récupère la liste des événements depuis le service. Il boucle sur les messages et donne les données d'événements au composant Event grâce aux Inputs.
- Composant Fils: Event: Le composant affiche les données qu'il récupère en Inputs et les affiche.

Une fois que votre composant fils (event) affiche les données reçues depuis le père (la liste), on veut établir la communication du père vers le fils. Lorsque l'utilisateur clique sur événement, on veut le rediriger vers une page seule contenant le détail complet de l'événement.

Il nous faut donc récupérer le "click" et le faire remonter au composant père. Pour faire ça, on utilise des Outputs. Il faut que le composant Event envoie les données de l'événements sur lequel l'utilisateur vient de cliquer afin de pouvoir le rediriger sur la bonne page.

En résumé:



Step 5 : Routing: Navigation

Maintenant que le composant “liste d'événement” réagit au changements dans les composants fils, on veut pouvoir ouvrir une page pour avoir plus de détail sur l'événement.

Pour commencer avec le routing, vous pouvez lire la page sur les routes en Angular. Le routing correspond à la navigation sur une application.

Dans le starter, vous trouverez un fichier routes.ts dans lequel on lie une route à un composant. Dans app.component.html, vous avez un exemple de route avec un lien (balise <a>) et la l'élément du DOM dans lequel le composant MessageList va s’afficher (balise <router-outlet>)

```
<a routerLink="messages"> Messages </a>

<router-outlet></router-outlet>
```

Dans notre cas, nous voulons préciser l’id de l'événement que l’on veut afficher et lancer la navigation directement à l’intérieur du composant Event list. Vous pourrez trouver toutes les informations ici.

POLYTECH
NICE SOPHIA

Côté serveur

Step 1 : récupération et installation du code

Cloner le repository github :

```
git clone https://github.com/delmotte/nodejs-starter-3a.git
```

Et installer les dépendances :

```
npm install
```

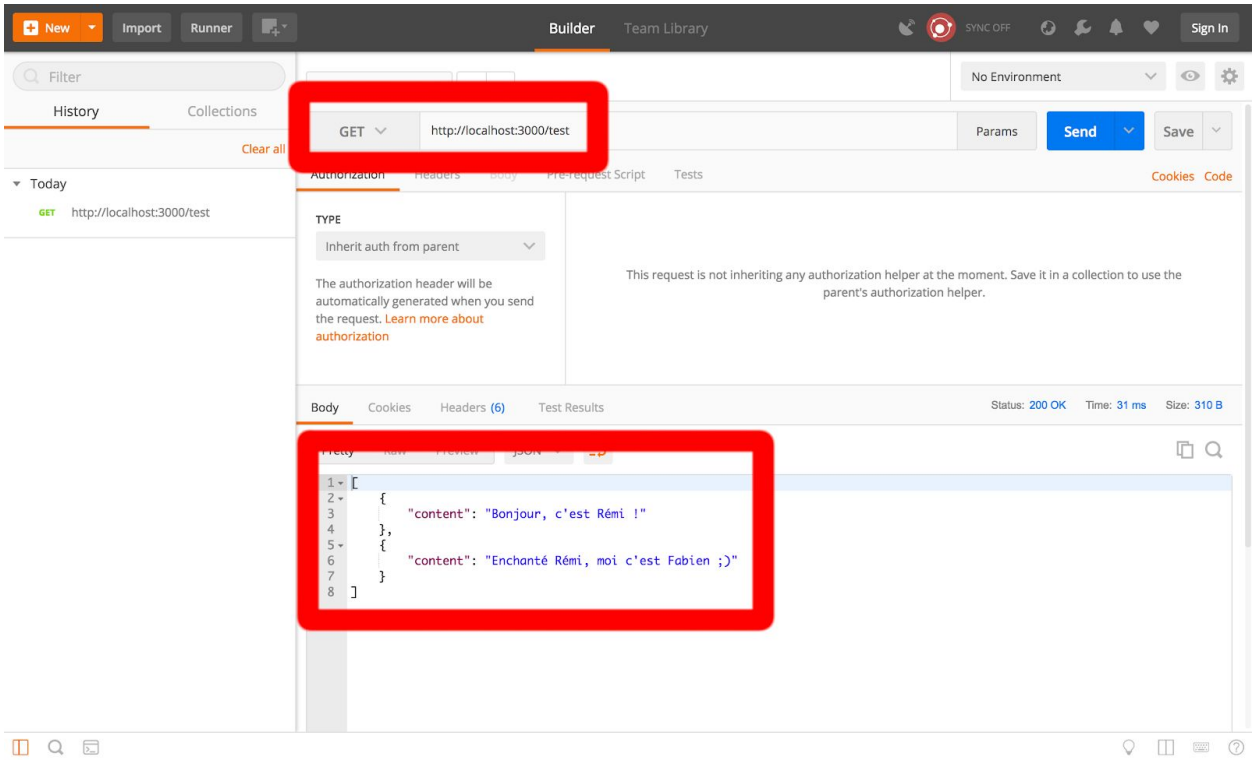
Il ne reste plus qu'à lancer le serveur :

```
node index.js
```

Step 2 : testez que tout fonctionne

Téléchargez Postman : https://www.getpostman.com/
Ouvrez le et effectuez une requête sur http://localhost:3000/test
Vous devriez obtenir le résultat suivant :

POLYTECH
NICE SOPHIA



Step 3 : prenez le temps de comprendre ce que fait le code

Oui, vous avez bien lu, et en plus il n'y a qu'un fichier de 27 lignes (index.js) !
Et changez le nom de la route (vous comprendrez quand vous aurez lu le fichier).

Step 4 : intégrez votre base de données

Il est venu le temps de remplacer le code à l'intérieur app.get(...) (Lignes 18 à 23) par un appel à votre base de données qui récupère les évènements !

Voilà les modules conseillés pour effectuer cette étape :
Postgres : https://www.npmjs.com/package/pg
mysql : https://www.npmjs.com/package/mysql

Step 5 : créez un évènement

Il ne reste plus qu'à créer une route pour créer un évènement : petit indice, c'est certainement un ***app.post*** qu'il faut faire.
Pour en savoir plus sur comment fonctionne **express**, nous vous invitons à consulter la documentation : http://expressjs.com/en/4x/api.html

POLYTECH
NICE SOPHIA

Pour aller plus loin

- ➔ Faites un PUT pour mettre à jour un évènement
- ➔ Faites un DELETE pour supprimer un évènement
- ➔ Améliorer votre UX (check des erreurs lors de l'édition, champ non vide par exemple, modale de confirmation lors d'un Delete)