Anthony RIMET: IUT Nancy-Charlemagne 2024

TD2 - Vue - Component

Introduction

Dans ce TD vous allez adapter l'application "FilmsDB" en utilisant les composants.

Pré-requis

Reprenez le code du TD2.

Exercice 1 : Historique

1. Créez un composant `History` qui affiche la liste des films consultés. Pour cela reprenez le code de votre précédant TD et adaptez le pour qu'il soit un composant. Placez le dans le dossier `components`.

2. Le composant attendra 2 props: `searches` et `currentSearch`. `searches` est un tableau de chaînes de caractères représentant les recherches effectuées. `currentSearch` est une chaîne de caractères représentant la recherche actuelle. Vous pouvez ajouter une vérification sur les types.

3. Au click sur un élément de la liste, le composant doit afficher la recherche correspondante. Pour ça vous allez devoir utiliser les événements. Créez un événement `show-search-results` qui permettra au composant parent d'appeler la méthode `showSearchResults` avec la recherche en paramètre.

Exercice 2 : CurrentMovie

1. Créez un composant pour afficher le film selectionné. Concernant la computed `duration` vous pouvez la déplacer dans le composant.

Exercice 3 : Search Bar

1. On veut placer le Titre de l'application, ainsi que l'input de recherche dans un composant `SearchBar`. Ce composant attendra une prop `title` qui est une chaîne de caractères. Ce composant doit être créé comme un "champ de formulaire" avec la possibilité de lier un `v-model` (currentSearch). N'oubliez pas de checker la documentation si besoin.

Exercice 4 : Results

1. Créez le composant ResultsList. Le composant doit attendre deux props : items (de type Array, obligatoire) et nbItems (de type Number, obligatoire).

2. Validez les props :

- `items` est un tableau (Array) et est obligatoire.

- `nbItems` est un nombre (Number) et est obligatoire. (attention : il ne s'agit pas du nombre d'items retournés par la requête ajax dans data.Search, mais du nombre total d'éléments possibles, fourni également par l'API OMDB)

3. Créez le template avec deux sections dans le template :

- Un slot scoped nommé total, qui fournit au parent le nombre total d'éléments.
- Un slot scoped par défaut avec un v-for pour parcourir et afficher chaque item de la liste, avec un attribut key unique.

4. Importer `ResultsList` dans le fichier principal de l'application.

- Remplacer le contenu actuel par le composant .
- Passer les props items et nbItems dans la balise .
- Utiliser le slot nommé total pour afficher le nombre de résultats.
- Ajouter le template pour le slot par défaut, en réutilisant le code précédent pour afficher chaque film de la liste à partir des données fournies par le slot.

Exercice 5 : Pagination

1. Ajoutez à la fin du composant `ResultsList` une div clickable pour passer à la page suivante. Cette div doit être conditionnelle et n'apparaître que si le nombre de résultats est inférieur au nombre total d'éléments possibles.

2. Créez un event `list:more` qui permettra au composant parent de charger plus de résultats.

3. Dans le composant parent, ajoutez une méthode `fetchMoreMovies` qui permettra de charger plus de résultats.

4. Cette méthode doit réaliser un appel Ajax sur la prochaine "page" de résultats, selon les mots- clés de la recherche courante. Étudiez la documentation de l'API pour savoir comment obtenir les résultats suivants. Tips : vous aurez besoin d'une data pour stocker la page courante. Pensez à la réinitialiser quand vous nettoyez toutes vos recherches

5. Mergez les résultats avec les résultats déjà affichés. La pagination devrait ainsi fonctionner

---

On pourrait se demander l'intérêt de ce composant sachant qu'on a dû conserver les éléments graphiques des items de la liste, ainsi que le message indiquant le nombre total de résultat, dans le template du parent. Regardez bien la tête de votre composant ResultsList. Est-ce que vous voyez que vous venez de développer un composant que vous pourrez réutiliser dans d'autres circonstances, avec d'autres listes, peu importe leur contenu ? C'est l'intérêt des composants et des slots : ils vous permettent de développer des "boîtes à outils" réutilisables. Vous gardez la main dans le parent sur la façon dont les éléments sont a chés, mais vous bénéficiez d'une logique systématique (ici l'affichage d'une liste avec pagination).