Programmation web sur client

TD4 - AJAX

Matthieu Nicolas

Préambule

Toujours dans le but de m'aider à mieux m'organiser, on poursuit le développement de l'application web permettant de gérer une liste de tâches à réaliser.

Cette fois-ci, j'ai mis en place un serveur me permettant de centraliser la liste des tâches et de les conserver de manière durable. Dans le cadre de ce TP, nous travaillerons sur un serveur local ne conservant les tâches qu'en mémoire pour des raisons de simplicité. Sauf erreur de ma part, vous ne devriez pas avoir besoin de modifier le fichier `server.js` qui contient le code du serveur.

Le but du client va donc être de permettre à l'utilisateur de consulter les tâches précédemment créées, d'en ajouter de nouvelles et d'en supprimer.

Pour démarrer le serveur localement, utiliser la commande `npm start`.

Exercice 1 - Ajout d'un identifiant aux tâches

Afin d'identifier les tâches de manière plus sûre, nous ajoutons une propriété `id` aux instances de la classe `Todo`. Pour représenter l'`id`, nous utiliserons un simple entier que nous incrémenterons à la génération de chaque nouvelle tâche.

Dans le fichier `/public/models/todo.js`,

1. Ajouter le paramètre `id` en tant que 1er paramètre du constructeur de la classe `Todo`. Ce paramètre sera utilisé pour initialiser la propriété `id` des instances de `Todo`.

Exercice 2 - Récupération des tâches existantes

Le serveur met à disposition la route `/todo`. Cette route permet de récupérer la liste des tâches existantes, au format JSON, par le biais d'une simple requête GET.

Dans le fichier `/public/main.js`,

1. Effectuer une requête GET sur l'url `/todo/`. Récupérer le contenu de la réponse et afficher le dans la console pour vérification.

2. En reprenant votre code du TP précédent, modifier le DOM de la page web pour afficher à l'utilisateur la liste des tâches.

3. Modifier votre fonction `todoAsHTML()` pour ajouter l'identifiant de la tâche au code HTML généré. Pour cela, nous l'ajouterons en tant que `data-attribute`. Voir `https://developer.mozilla.org/en-US/docs/Learn/HTML/Howto/Use_data_attributes` pour plus d'informations à ce sujet.

Exercice 3 - Suppression d'une tâche complétée

Le serveur met à disposition la route `/todo/:id`, où `:id` est l'identifiant d'une tâche donnée. Cette route permet de supprimer la tâche associée à l'identifiant spécifié par le biais d'une simple requête DELETE.

Dans le fichier `/public/main.js`,

1. Ajouter un `EventListener` à la liste des tâches. Cet `EventListener` est déclenché lorsqu'un clic est effectué sur une tâche. Lors de son déclenchement, l'`EventListener` récupère l'identifiant de la tâche concernée et effectue la requête DELETE correspondante. À la complétion avec succès de la requête, retire la tâche de l'interface utilisateur.

Exercice 4 - Ajout d'une tâche

Le serveur met à disposition la route `/todo`. Cette route permet de créer une nouvelle tâche par le biais d'une requête POST. Cette requête doit fournir comme contenu une instance de `Todo`, au format JSON. L'objet envoyé doit donc posséder un `id`, un entier unique, un `title`, une chaîne de caractères non-vide et `deadline`, une date valide.

Dans le fichier `/main.js`,

1. Ajouter un `EventListener` au formulaire de création de tâche. Cet `EventListener` est déclenché lorsque le formulaire est validé. Lors de son déclenchement, l'`EventListener` instancie un nouveau `Todo` à partir des informations du formulaire et transfère cette objet au serveur au format JSON par le biais d'une requête POST. À la complétion avec succès de la requête, ajoute la tâche à l'interface utilisateur.