Vue.js : Interceptors & Guards

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en programmation par contrainte (IA)
Ingénieur en génie logiciel

elmouelhi.achref@gmail.com

Plan

1. Interceptors
2. `jwt-decode`
3. Guards
   - Guards globales
   - Guards locales

Vue.js

Hypothèse

- Les ressources **REST** utilisées dans les chapitres précédents sont désormais protégées
- Ces ressources nécessitent une authentification et un jeton **JWT**

Vue.js

Hypothèse

- Les ressources **REST** utilisées dans les chapitres précédents sont désormais protégées
- Ces ressources nécessitent une authentification et un jeton **JWT**

Comment ?

- Créer une interface d'authentification pour récupérer un jeton **JWT**
- Intercepter les requêtes externes pour attacher le jeton

Vue.js

Le template de `AuthView.vue`

```
<template>
    <h1>Authentification</h1>
    <form @submit.prevent="seConnecter">
        <div>
            <label for="username">Nom d'utilisateur</label>
            <input type="text" v-model="user.username" id="username">
        </div>
        <div>
            <label for="password">Mot de passe</label>
            <input type="password" v-model="user.password" id="password
               ">
        </div>
        <button>Se connecter</button>
    </form>
</template>
```

Et maintenant le `script`

```
<script setup>
import axios from 'axios'
import { reactive } from 'vue'
import { useRouter } from 'vue-router'

const router = useRouter();
const user = reactive({ username: '', password: '', grantType: '
  password' })
const seConnecter = () => {
    axios
        .post('http://localhost:3000/ws/token', user)
        .then(res => {
            localStorage.setItem('token', res.data.accessToken)
            router.push({ name: 'personne' })
        })
        .catch(err => alert(err))
}
</script>
```

Et maintenant le `script`

```
<script setup>
import axios from 'axios'
import { reactive } from 'vue'
import { useRouter } from 'vue-router'

const router = useRouter();
const user = reactive({ username: '', password: '', grantType: '
  password' })
const seConnecter = () => {
    axios
        .post('http://localhost:3000/ws/token', user)
        .then(res => {
            localStorage.setItem('token', res.data.accessToken)
            router.push({ name: 'personne' })
        })
        .catch(err => alert(err))
}
</script>
```

N'oublions pas d'associer une route `/auth` à ce composant dans `routes/index.js`

Vue.js

Ensuite

- Créons un fichier `axios-request.interceptor.js` dans `interceptors` (à créer aussi)
- Si on trouve le jeton dans le `localStorage`, alors on l'attache à l'entête de la requête sortante.

Vue.js

Contenu de `axios-request.interceptor.js`

```
export function axiosInterceptor(config) {
    if (localStorage.getItem('token') != null
        && localStorage.getItem('token') != undefined) {
        const token = localStorage.getItem('token');
        config.headers.setAuthorization(`Bearer ${token}`)
    }
    return config
}
```

Vue.js

Il suffit maintenant d'utiliser cette fonction dans `main.js`

```
import { createApp } from 'vue'
import App from './App.vue'
import router from './router'
import '@/validators/min-max'
import axios from 'axios'
import VueAxios from 'vue-axios'
import { axiosInterceptor } from '@/interceptors/axios-request.interceptor'

const app = createApp(App);

axios.interceptors.request.use(axiosInterceptor)

app
    .use(router)
    .use(VueAxios, axios)
    .mount('#app')
```

Vue.js

jwt-decode

- Package **Node.js**
- Écrit en **TypeScript**
- Permettant de décoder un jeton **JWT** et retourner un objet **JavaScript**
- `https://www.npmjs.com/package/jwt-decode`

Vue.js

Pour l'installer

```
npm install jwt-decode
```

Vue.js

Exemple

```
var decoded = jwt_decode(token);

console.log(decoded);
```

Vue.js

Hypothèse

- Tous les composants doivent être accessible uniquement aux utilisateurs authentifiés
- Tous les utilisateurs non-authentifiés seront redirigés vers la page d'authentification

Vue.js

Hypothèse

- Tous les composants doivent être accessible uniquement aux utilisateurs authentifiés
- Tous les utilisateurs non-authentifiés seront redirigés vers la page d'authentification

Comment ?

Utiliser une `guard`

Vue.js

Dans `routes/index.js`, définissons notre `guard` qui s'exécutera avant chaque requête

```
// le contenu précédent

router.beforeEach((to, from, next) => {

})
// il faut le définir avant de l'exporter
export default router
```

Vue.js

Dans `routes/index.js`, définissons notre `guard` qui s'exécutera avant chaque requête

```
// le contenu précédent

router.beforeEach((to, from, next) => {

})
// il faut le définir avant de l'exporter
export default router
```


Explication

- `to` : données sur la route que l'utilisateur souhaite visiter
- `from` : données sur la route actuelle
- `next` : permet de continuer à faire la suite

Vue.js

Autres triggers

- `afterEach`
- `beforeRouteLeave`
- `beforeRouteUpdate`
- ...

Vue.js

Redirigeons toutes les requêtes venant d'un utilisateur non-authentifié à la page d'authentification

```
router.beforeEach((to, from, next) => {
  if (to.path != '/auth' &&
  (localStorage.getItem('token') == null ||
  localStorage.getItem('token') == undefined)) {
      router.push({ name: 'auth' })
  }
  next()
})
```

Vue.js

Remarque

On peut installer une guard sur une ou quelques route·s (mais pas toutes)

Vue.js

Remarque

On peut installer une guard sur une ou quelques route·s (mais pas toutes)

N'oublions pas de commenter le code de la guard globale `beforeEach`

Vue.js

Sécurisons uniquement l'accès à la route `/personne`

```
{
    path: '/personne',
    name: 'personne',
    component: PersonneShowView,
    beforeEnter: (to, from) => {
        return (localStorage.getItem('token') != null &&
                localStorage.getItem('token') != undefined)
    }
}
```