POLYTECH MARSEILLE Aix*Marseille Université **Informatique – 4A**

Applications web et mobiles

TP n°12 : Backend en Node/Express – la finale

Étape 56 – Authentification dans le backend en Node

En PHP, l'authentification se faisait via des sessions PHP. Ici, on va utiliser le même principe : cookies de session. Cela nous amène à un mécanisme appelé JSON Web Token (JWT). Mais, en Node, ces cookies sont gérés un peu différemment : contrairement à Apache/PHP qui conserve les informations de session sur le serveur, en Node, celles-ci ne sont pas conservées sur le serveur et doivent donc être renvoyées à chaque requête du client (Angular/votre navigateur). Cela permet de « passer à l'échelle » et de gérer de nombreux clients.

Afin d'exploiter les JWT, dans le répertoire de votre backend, installez via `npm` les packages suivants : `jsonwebtoken` et `cookie-parser`. Afin que votre instance `app` d'`express` utilise `cookie-parser`, qui permet d'exploiter aisément les cookies dans les applications Node, il vous suffit de rajouter les deux lignes suivantes dans `app.js` :

```
const cookieParser = require('cookie-parser');
app.use(cookieParser());
```

Récupérez maintenant les fichiers suivants :

https://pageperso.lis-lab.fr/christophe.gonzales/teaching/mobile/ressources/fr-FR/sessionJWT.js
https://pageperso.lis-lab.fr/christophe.gonzales/teaching/mobile/ressources/fr-FR/auth.js

Le premier permet de créer des cookies de session JWT ainsi que de récupérer les informations contenues dans ces cookies. Son code est commenté de manière à ce que vous puissiez comprendre comment on exploite les JWT. Puisque ces cookies ont vocation à authentifier les utilisateurs, le reste de votre backend n'interagit pas directement avec `sessionJWT.js` mais plutôt avec `auth.js` qui, lui, exploite `sessionJWT.js`, comme le montrent les fichiers du backend de démonstration.

Maintenant que vous avez l'équivalent de la notion de session PHP, vous pouvez réécrire en Node l'équivalent de votre fichier `checkLogin.php`, à savoir un fichier `checkLogin.js`. Les fonctions de `auth.php` utilisées dans `checkLogin.php` ont vocation à être retranscrites en JavaScript dans le fichier `auth.js` et le code écrit dans `checkLogin.php` doit être retranscrit en JavaScript dans le fichier `checkLogin.js`. La seule vraie différence avec le PHP réside dans le fait qu'il faut explicitement envoyer au client le cookie de session avant de lui envoyer le message JSON. Dans le fichier `checkLogin.js`, cela peut être réalisé en appelant `auth.sendSessionCookie (req, res, {userId: id})`, où `id` est l'identifiant de l'utilisateur qui a réussi à se connecter. Si vous regardez le code de la fonction `sendSessionCookie`, vous verrez que cette fonction s'attend à ce que son 3ème paramètre soit une session. Normalement, une session contient 3 informations : un `userId` et deux champs correspondant à des dates d'expiration. Si le cookie a expiré, on en recrée un nouveau (mais c'est coûteux en temps de calcul), sinon on réutilise

le cookie existant. Dans le fichier `sessionJWT.js`, j'ai paramétré des expirations au bout de 30mn. Ainsi, la génération des cookies ne sature pas inutilement le CPU du serveur backend. Si vous passez comme session l'objet `{userId: id}`, il manque les informations sur les expirations et, dans ce cas, la fonction `sendSessionCookie` du fichier `sessionJWT.js` crée automatiquement un nouveau cookie. Dans les autres routes que celle du `checkLogin`, si vous regardez le code correspondant (par exemple dans la fonction `getCours1` du fichier `getCours1.js` du backend de démonstration), vous verrez que la première instruction consiste à récupérer la session courante. Par conséquent, cette session peut, par la suite, être utilisée dans l'instruction `auth.sendSessionCookie (req, res, session)`, comme le montre le code de la fonction `getCours1`.

Testez votre système d'authentification avec votre application *Angular*.

Étape 57 – Tests finaux

Normalement, votre application *Angular* fonctionne complètement. Toutes les vues (pages) doivent à nouveau fonctionner correctement. Testez le. Si tout est ok, votre backend et votre application *Angular* sont terminées.

Optionnel : pour ceux/celles qui souhaitent en savoir plus

Étape 58 – Comprendre les JWT

L'url https://hackernoon.com/why-do-we-need-the-json-web-token-jwt-in-the-modern-web-k29l3sfd décrit très clairement comment fonctionnent les JWT et les raisons pour lesquelles les informations de session ne doivent pas être stockées sur le serveur.

Les JWT exploitent un mécanisme de cryptographie par clef publique/privée (l'URL suivante montre comment générer ces clefs : https://gist.github.com/ygotthilf/baa58da5c3dd1f69fae9).

La ressource suivante peut également vous être utile pour comprendre le fonctionnement d'une authentification par JWT : https://blog.angular-university.io/angular-jwt-authentication/.