Cave à vins (Partie 13)

# TP02 du module 06 – Sécurité

|  |
| --- |
| Ce TP sera un TP fil rouge du cours   * Transfert d’un jeton JWT entre le client et le serveur * Mise en place des librairies JJWT * Création de la configuration associée |

|  |
| --- |
| **Durée estimée** |
| 1 heure 30 |

Contexte

En vous appuyant sur l’ensemble des concepts abordés dans le cours, mettez en place une application Spring Boot permettant la gestion d’une cave à vins.

L’application de cave à vin sera une application d’API Web sécurisée à la fin des itérations :

* Elle permet de gérer le stock des bouteilles de vins par le propriétaire qui est un administrateur de l’application
* Elle permet de gérer l’achat en ligne des bouteilles par un client
  + Les bouteilles de vin peuvent avoir 3 couleurs : Rouge, Blanc ou Rosé
  + Les bouteilles de vin sont associées aux régions françaises
* Un client et un propriétaire doivent s’authentifier
* Un visiteur est un client qui n’a pas de compte
* Il pourra uniquement consulter le stock

Dans les itérations nous réaliserons uniquement le back de l’application.

Cette application se fera avec Spring Boot.

Dans les itérations de ce module, nous allons sécuriser notre API Web.

Enoncé

Pour réaliser ce TP, il faut vous aider de la démonstration et des schémas du cours

* Cycle d’authentification

Une image contenant texte, capture d’écran, Police, diagramme

Description générée automatiquement

* Cycle d’autorisation

Une image contenant texte, Police, capture d’écran, diagramme

Description générée automatiquement

# Librairies JJWT :

Il faut ajouter les lib librairies JJWT API et JJWT Impl pour permettre de manipuler le jeton JWT

* Dans build.gradle ; ajouter les librairies pour JWT

# L’entité Utilisateur :

Dans les 2 schémas, il faut manipuler les données d’un utilisateur depuis la base. Pour cela, il faut une entité et un Repository associé

* L’entité correspond à la table « cav\_user » déjà existante

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

* L’entité Utilisateur doit implémenter l’interface UserDetails

# UserDetailsService :

Pour manipuler UserInfo avec Spring Security ; il faut déclarer UserDetailsService

* UserDetailsService doit être déclaré dans une classe de configuration
* Création d’une classe JwtAppConfig

# Créer JwtService

D’après les 2 schémas, JwtService va gérer le jeton JWT.

* C’est pour manipuler le jeton JWT, qu’il fallait ajouter les dépendances JJWT API et JJWT Impl

# Créer JWTAuthenticationFilter

Si on regarde notre schéma, l’élément qui intercepte les requêtes est JwtAuthenticationFilter

* Elle hérite d’OncePerRequestFilter
  + Comme son nom l’indique ; il sera utilisé à chaque appel
* Elle doit manipuler 2 services
  + JwtService 🡪 qui gèrera le jeton JWT
  + UserDetailsService est une interface de Spring Security qui permet de manipuler l’utilisateur remonté de la base de données

# Contexte de Spring Security

D’après nos schémas :

* Activer le JwtAuthenticationFilter comme premier filtre avec celui du contexte de sécurité
* Définir la solution de reconnaissance de l’utilisateur à connecter (JwtAppConfig)

# Contrôleur et service d’authentification

D’après notre schéma d’authentification, il nous reste à créer : AuthenticationController et AuthenticationService

* Le contrôleur AuthenticationController permettra de récupérer les données d’authentification
  + L’URL racine déclaré ; dans la classe AppConfigSecurity : /caveavin/auth/\*\*
  + Il faut qu’il manipule la base de données, au travers du service AuthenticationService
* Il déclare une méthode sur Post pour gérer l’authentification.
* Pour cette méthode il faut déclarer
  + AuthenticationResponse 🡪 POJO contenant uniquement le jeton JWT à retourner
  + AuthenticationRequest 🡪 POJO représentant les données du formulaire de connexion

# Exécution :

Pour manipuler notre sécurité avec jeton JWT

1. Créer un jeton JWT en connectant un Utilisateur avec le rôle CLIENT
2. Créer un jeton JWT en connectant un Utilisateur avec le rôle OWNER
3. Copier les jetons JWT pour manipuler les URLs de l’application