SIMPLON.CO

Jobintech

Documentation technique du projet LabXpert - Spring Sécurité(Part 3)

Réalisé par

GUELSA Mouna

2023-2024

TABLE DES MATIÈRES

TABLE DES FIGURES

1 Introduction

Dans tout projet basé sur Spring, garantir la sécurité des données et des utilisateurs est primordial. Intégrer des mesures de sécurité solides dès le début du développement est essentiel pour prévenir les vulnérabilités et assurer la confiance des utilisateurs. Dans cette optique, l'utilisation de Spring Security s'avère être une solution efficace et intégrée pour gérer l'authentification, l'autorisation et d'autres aspects de la sécurité dans votre application. Cette introduction explorera les principaux points à considérer lors de l'ajout de la sécurité à votre projet Spring, mettant en lumière l'importance de cette démarche et les avantages de l'utilisation de Spring Security.

2 Etude technique et technologique

*** Spring Securité :**

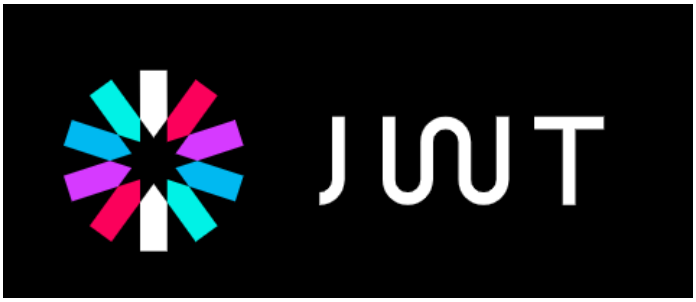
Spring Security est un puissant framework d'authentification et de contrôle d'accès pour les applications Java, notamment celles construites avec le framework Spring. Il offre des services de sécurité complets pour les applications logicielles d'entreprise basées sur Java EE. L'objectif principal de Spring Security est de sécuriser les applications Java en fournissant une authentification robuste, une autorisation et d'autres fonctionnalités de sécurité.

Les principales fonctionnalités de Spring Security incluent :

1. **Authentification et Autorisation** : Spring Security permet aux développeurs d'authentifier les utilisateurs à partir de différentes sources telles que LDAP, les bases de données, OAuth, etc. Il offre également des capacités d'autorisation fines, permettant aux développeurs de contrôler l'accès à des ressources spécifiques en fonction des rôles ou des permissions des utilisateurs.
2. **Intégration avec le Framework Spring** : Comme Spring Security est construit sur le framework Spring, il s'intègre parfaitement avec les autres composants Spring, ce qui facilite l'incorporation des fonctionnalités de sécurité dans les applications basées sur Spring.
3. **Sécurité Web** : Spring Security offre un support robuste pour sécuriser les applications Web, y compris des fonctionnalités telles que la protection CSRF, la gestion des sessions et le support de divers mécanismes d'authentification Web (basés sur formulaire, HTTP Basic, HTTP Digest, etc.).
4. **Sécurité au Niveau des Méthodes** : Les développeurs peuvent sécuriser des méthodes individuelles ou des composants de la couche de services à l'aide d'annotations fournies par Spring Security, permettant un contrôle d'accès fin au sein de l'application.
5. **Sécurité des API RESTful** : Spring Security offre un support pour sécuriser les API RESTful en utilisant des mécanismes tels que OAuth 2.0, JWT (JSON Web Tokens) et d'autres schémas d'authentification et d'autorisation adaptés aux services RESTful.

6. **Authentification Remember Me** : Il prend en charge l'authentification "Remember Me", permettant aux utilisateurs de contourner le processus de connexion après s'être authentifiés une fois avec succès, généralement en utilisant des cookies persistants.
7. **Intégration avec d'Autres Standards de Sécurité** : Spring Security s'intègre parfaitement avec d'autres normes et protocoles de sécurité tels que OAuth, OpenID, SAML, etc., permettant aux développeurs de construire des applications sécurisées conformes aux normes de l'industrie.

*** JWT**



JSON Web Tokens (JWT) est un standard ouvert (RFC 7519) qui définit une manière compacte et auto-contenue de transmettre de l'information de manière sécurisée entre deux parties, sous forme d'un objet JSON. Ces informations peuvent être vérifiées et sont donc fiables en raison de la signature numérique qu'elles portent. Les JWT sont souvent utilisés pour l'authentification et l'échange sécurisé d'informations entre différentes parties d'une application.

Voici comment fonctionne un JWT de manière simplifiée :

1. **Création du Token** : Lorsqu'un utilisateur se connecte à une application, le serveur génère un JWT qui contient les informations d'identification de l'utilisateur (par exemple, l'identifiant de l'utilisateur, son rôle, etc.). Ce JWT est signé à l'aide d'une clé secrète connue uniquement par le serveur.
2. **Transmission du Token** : Le JWT est renvoyé au client (généralement stocké dans un cookie ou dans le corps d'une réponse HTTP). À partir de ce moment, le client envoie ce JWT dans l'en-tête d'autorisation de chaque requête ultérieure.
3. **Validation du Token** : Lorsque le serveur reçoit une requête avec un JWT dans l'en-tête d'autorisation, il peut valider ce token en vérifiant sa signature à l'aide de la clé secrète. Si la signature est valide, le serveur extrait les informations qu'il contient pour authentifier l'utilisateur et autoriser ou non la demande.
4. **Expiration et Renouvellement** : Les JWT peuvent avoir une durée de vie limitée, définie par une expiration. Après expiration, le JWT n'est plus valide et l'utilisateur doit se reconnecter pour obtenir un nouveau JWT.

Les avantages des JWT incluent leur compacité, leur auto-contenue, et leur capacité à être facilement transmis via des mécanismes tels que les en-têtes HTTP. Cependant, comme ils sont auto-signés, les JWT ne sont pas révoquables avant leur expiration, à moins de maintenir une liste noire des tokens invalides.

*** OAuth 2.0**

OAuth 2.0 est un protocole d'autorisation ouvert et standard qui permet à une application tierce d'accéder aux ressources d'un utilisateur sans avoir besoin de ses identifiants d'authentification. Il est largement utilisé pour permettre l'accès sécurisé aux API Web et aux données

utilisateur dans de nombreux scénarios, tels que les applications mobiles, les applications web, les API RESTful, etc.

Voici les principaux acteurs dans le protocole OAuth 2.0 :

— **- Resource Owner (Propriétaire de la ressource)** : L'utilisateur final qui possède les données ou les ressources protégées.

— **- Client** : L'application tierce qui souhaite accéder aux ressources de l'utilisateur.

— **- Authorization Server (Serveur d'autorisation)** : Le serveur qui vérifie l'identité de l'utilisateur et délivre les tokens d'accès après avoir autorisé l'application tierce.

— **- Resource Server (Serveur de ressources)** : Le serveur qui héberge les ressources protégées que le client souhaite accéder.

Le protocole OAuth 2.0 définit plusieurs flux d'autorisation, notamment :

1. **Authorization Code Flow (Flux de code d'autorisation)** : Utilisé par les applications web serveur à serveur pour obtenir un token d'accès en échange d'un code d'autorisation.
2. **Implicit Flow (Flux implicite)** : Utilisé par les applications web clientes (par exemple, les applications JavaScript) pour obtenir directement un token d'accès.
3. **Client Credentials Flow (Flux de crédentials client)** : Utilisé par les applications qui agissent en tant que client pour accéder à leurs propres ressources, sans impliquer un utilisateur final.
4. **Resource Owner Password Credentials Flow (Flux de crédentials utilisateur)** : Utilisé lorsque le client demande directement les identifiants de l'utilisateur final et les utilise pour obtenir un token d'accès.

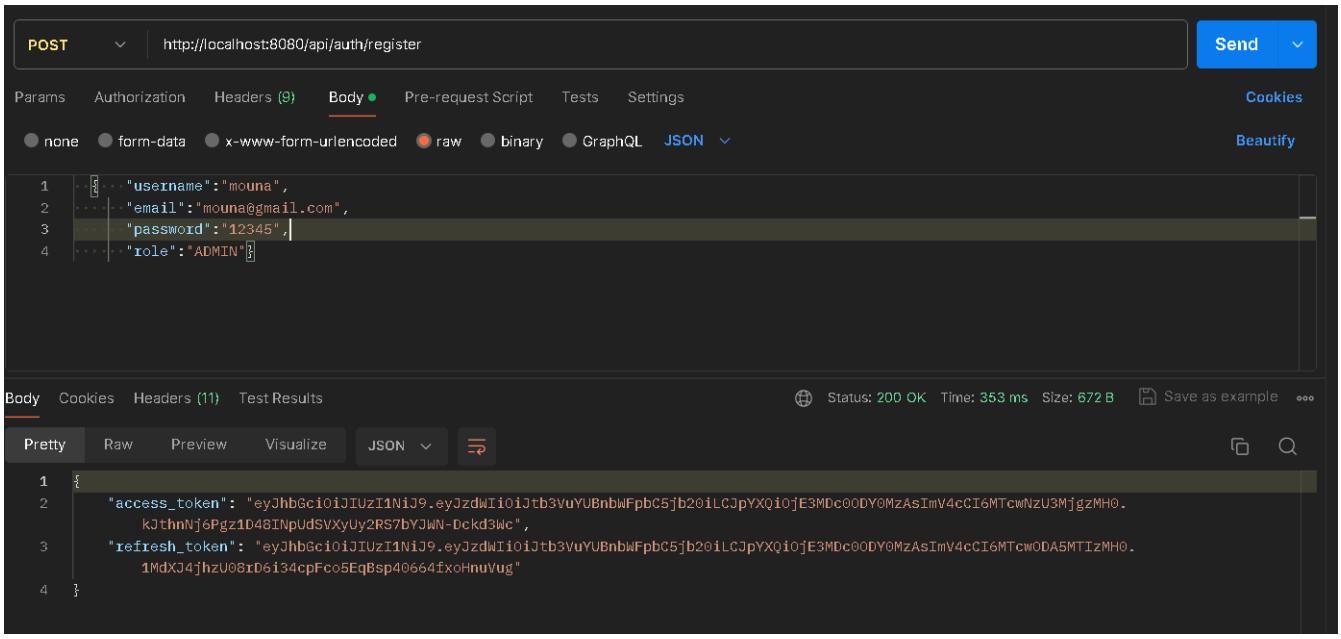
OAuth 2.0 est un protocole flexible et extensible, largement adopté par l'industrie pour son efficacité et sa sécurité dans la gestion des autorisations et de l'accès aux données utilisateur.

*** ¨Postman :**

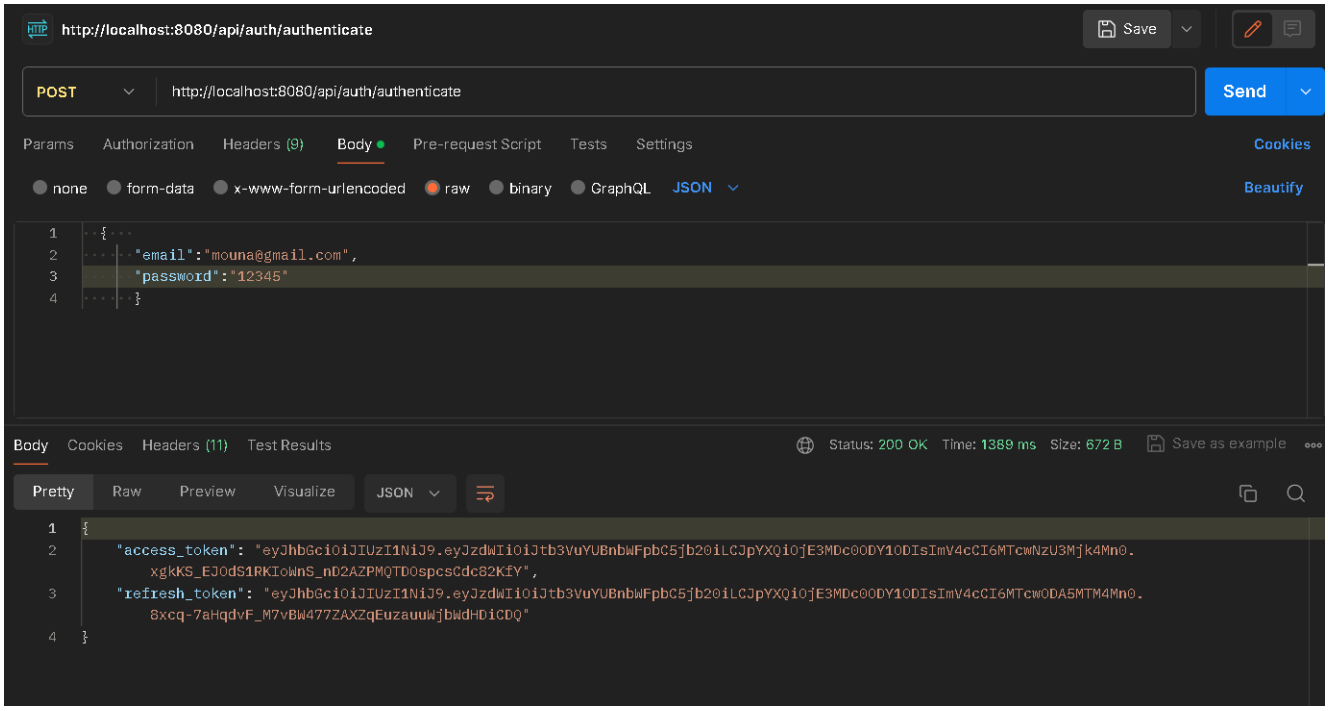
Postman est officiellement présentée comme une plateforme API pour la création et l'utilisation d'API. D'une manière générale, Postman est une plateforme qui permet de simplifier chaque étape du cycle de vie des API et de rationaliser la collaboration, afin de créer, plus facilement et plus rapidement, de meilleures API.Il permet de tester manuellement des des requ^etes HTTP.

3 Démonstration

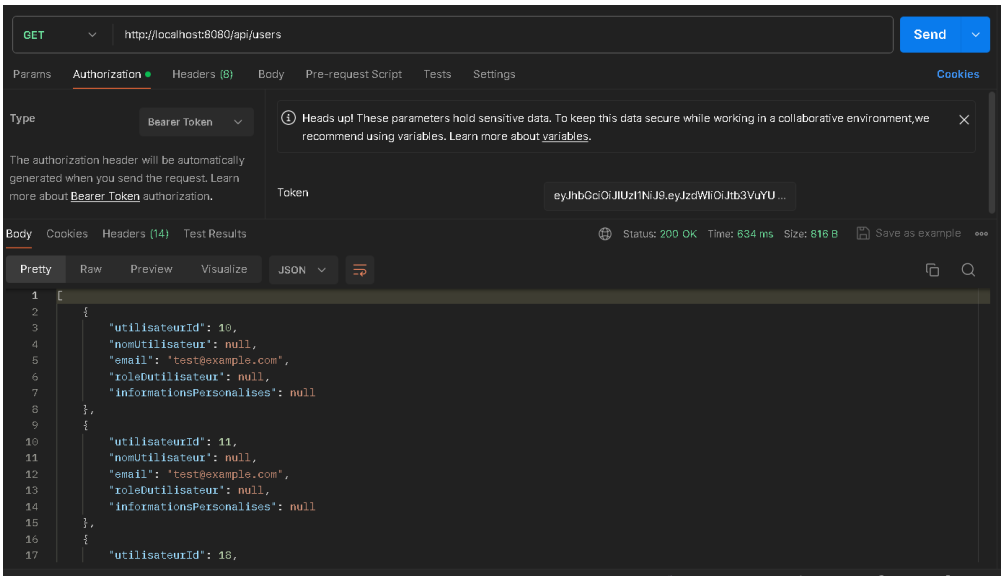
*** Register :**



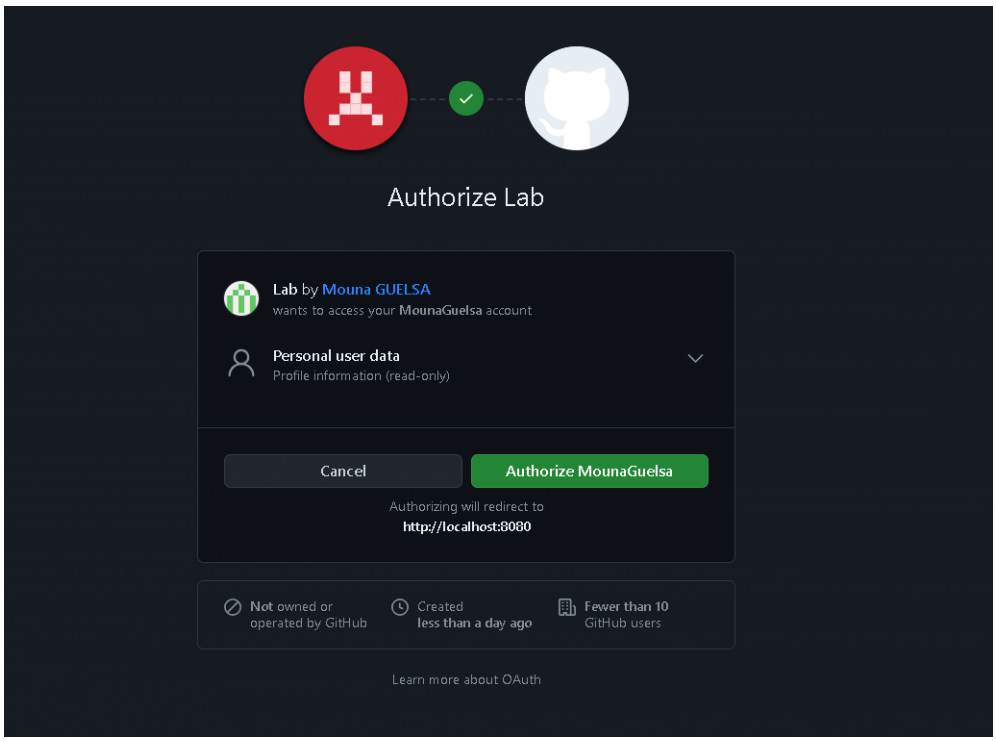
*** Authentification :**



*** Authorisation :**



*** Oauth 2.0 :**



4 Conclusion

Intégrer la sécurité à un projet Spring est une étape cruciale pour assurer la protection des données et la confiance des utilisateurs. En adoptant une approche proactive en matière de sécurité dès les premières phases du développement.Spring Security offre une solution complète et flexible pour répondre aux exigences de sécurité modernes, en simplifiant la gestion de l'authentification, de l'autorisation et d'autres aspects de la sécurité.