OIDC Implementation

Version: 0.1

Date: 13.06.2023

Table of Contents

Version History:

Version	Date	Author	Description of Changes
0.1	13.06.23	Jayasakthi Balaji G	Initial draft

Introduction/Objective:

The objective is to implement OpenId connect and Oauth2 for authentication and authorization through an example application.

Requirement Overview:

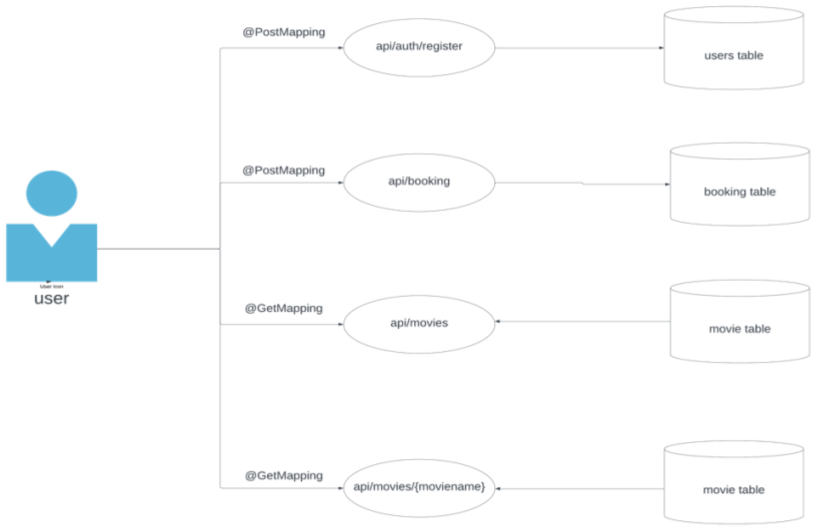
- The requirement mainly based on the roles specification and authorization after authentication of the roles, So this is done through a movie booking application.
- A ticket booking application have to be developed which is embedded with Oauth2 and Open Id connect.

Feature design

The main roles in this application are,

- User
- Operator

1. The below flow explains the resources that the user can authorize,



Operations of the User are,

@GetMapping

getMovies(): Can fetch all the movies from the theatre along with the details of movies, example, name, genre, showtime, director etc..,

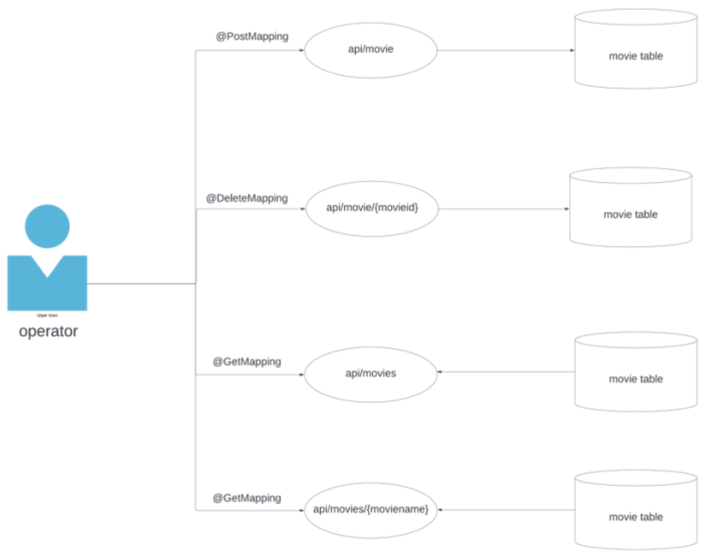
@GetMapping

getMovie(): Can fetch only all the movie names from the theatre

@PostMapping

@addBooking(): Can able to book a ticket for a movie which needs the basic information of the user.

2. The below flow explains the resources that the operator can authorize,



Operations of the Operator are,

@GetMapping

getMovies(): Can fetch all the movies from the theatre along with the details of movies, example, name, genre, showtime, director etc..,

@GetMapping

getMovie(): Can fetch only all the movie names from the theatre

@PostMapping

addMovie(): Can add a movie to the theatre in which the user can book tickets

@DeleteMapping

deleteMovie(): Can remove a movie from the theatre

Prerequisites of the application

- Login for authentication
- User and Operator for authorization

1. Dependencies to be added,

⇒ Spring security

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

⇒ Oauth2 client

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-oauth2-client</artifactId>
</dependency>
```

⇒ Oauth2 authorization server

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-oauth2-authorization-server</artifactId>
</dependency>
```

⇒ Spring web starters

```
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-web</artifactId>
</dependency>
```

⇒ Postgres

```
<dependency>
  <groupId>org.postgresql</groupId>
  <artifactId>postgresql</artifactId>
  <scope>runtime</scope>
```

⇒ Spring boot starter jpa

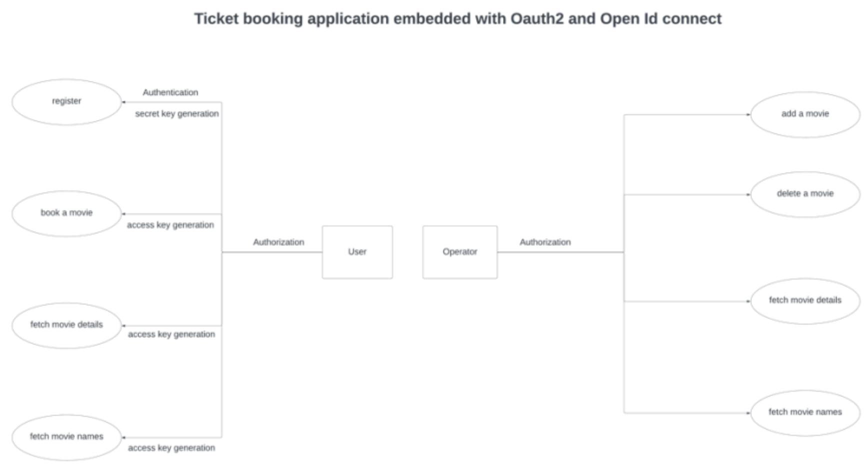
```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

⇒ Json web token

```
<dependency>
  <groupId>io.jsonwebtoken</groupId>
  <artifactId>jjwt-api</artifactId>
  <version>0.11.2</version>
</dependency>
```

2. Configurations in application.yml
    a. Necessary configurations for connecting the db
    b. Necessary configuration for the security (authentication and authorization)
3. Add controllers and endpoints
    a. @GetMapping
    b. @PostMapping
    c. @DeleteMapping
4. Start the application

Flow of the application



User role:

⇒ Let's take the role id of the user as 1,
⇒ When the starts the booking, the user needs to be an existing customer, so that the initial stage is registering, when the user registers himself a token will be generated, with the generated token the authentication happens to book a movie
⇒ Only if the user is a existing customer authentication happens and given all the authorization respective to the user role.

Operator role:

- Let's take the role id of the operator as 2,
- Operator can able to add a movie to the theatre (database), in which the user can book tickets according to the movie name specified by the operator.
- Operator can also delete a movie from the theatre (database).

⇒ Here the user's authentication works with the Open Id.
⇒ The authorization according to the role id is done with the Oauth2.

Information about Oauth2 and Open Id connect of the application

⇒ Open Id connect and Oauth2.
⇒ OIDC – Open Id connect.

Oauth2

- Oauth (open authorization) framework is a protocol is mainly used for **authorization** purpose (tells the user what to access)
- Oauth generates access tokens that is used for authorization
- Scopes: The permission or the authorization
- Oauth2 roles have specific scopes so that the roles could access the specific set of operations

Roles of Oauth2

- Resource owner
- Resource server
- Authorization server
- Client/ Application

Resource owner: Something who owns the information

Resource server: Something who holds the information

Authorization server: Locking or giving security to the information

Client/Application: Who asks request for the authorization

Open Id connect

⇒ Open Id connect is an extension to Oauth2 which is mainly used for the **authentication** purpose (tells only the user is valid or not)
⇒ Open Id generates id token used for authentication

Uses of OIDC

- Defines own roles and access
- Efficient to authenticate our own API's endpoints

Tokens

- Access Token
  - Bearer token
  - JWT
  - Opaque
- Refresh Token

Access token: Used for the authorization purposes [JWT will be used]

Refresh token: Used to generate new access tokens without the reauthentication of the users

Use of Oauth2 and Open Id when combined

- Stores user credentials
- Login security
- User registration management
- Integration of LDAP [Lightweight directory access protocol]
- Password reset process
- Two factor authentication

Cross platform authentication

User could use same login credentials to access multiple set of cross platforms applications.

Example: To host a static website in Netlify kind of page, which uses GitHub credentials to access the GitHub folders to host the website.

Grant Types

- Client credentials flow
- Authorization code flow
- Device code flow
- Refresh flow
- Password flow

Client credentials flow: Simple flow like the username/passwords flow but the client is not trusted here. Generates a secret key which gives the access keys to authorize certain operations.

Password flow: Simple flow, it has three layers user, client, server. Here client must be trusted, as client has the actual password which is insecure.

Authorization code flow: Its confidential, secure, browser based. Only the authenticated devices could access the information. User can approve the permissions to the devices for the authorization.

Reference Links:

1. Spring security Oauth Authorization server
2. Spring security and Open Id connect
3. JWT debugger
4. JWT intro and overview
5. Spring initializer