**Details:**

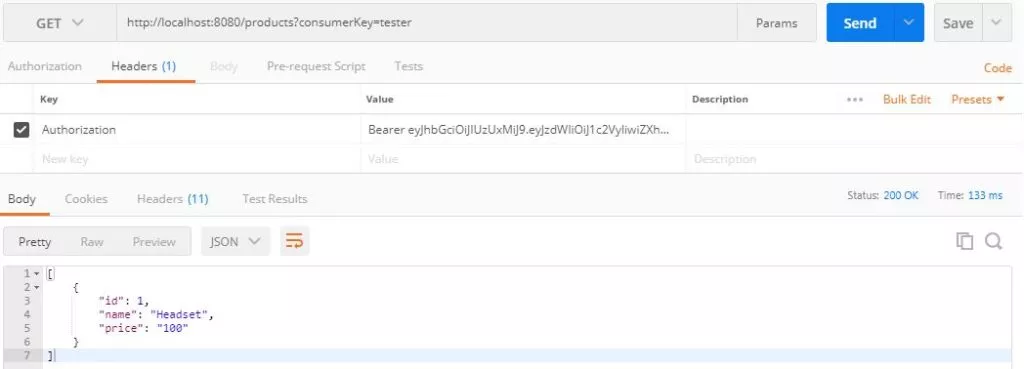
Technologies used:

1. Spring boot
2. Spring data JPA
3. H2 in memory database (For testing purposes. Can be changed to DB of choice.)
4. Swagger
5. Spring security
6. JSON web token
7. Lombok

For demonstration purposes an end point with /product is created to show role based access to the end point for create/read/update/delete/list

1. **Generating JWT** – Expose a POST API with mapping /authenticate. On passing correct username and password it will generate a JSON Web Token (JWT)
2. **Validating JWT** – If user tries to access Product API with mapping /product/\*\*, it will allow access only if the request has a valid JSON Web Token (JWT) and users with appropriate roles will be allowed to perform create/update/delete/list operations.

Pass the jwttoken appended with Bearer with key as Authorization when using outside swagger.



**Model design:**

1. User -> Stores the user details like username and password.
2. Role -> Stores the role name, api and access to api

Example:

role name : productreadrole

Api: product

ApiAccess : read

role name : productwriterole

Api: product

ApiAccess : write

1. UserRole -> Mapping between user and role
2. Product -> Demo entity to showcase access to Api.

**RestContoller:**

1. /product/list -> access to roles which has list access
2. /product/get/{productId} -> access to roles which has get access
3. /product/create -> access to roles which has write access
4. /product/update/{productId} -> access to roles which has update access
5. /product/delete{productId} -> access to roles which has delete access

**Steps to test flow with swagger:**

1. Create a user using /user/create API.
2. Create roles using /role/create API
   1. Alternatively you can create users along with roles.
3. Create association with /user/associateuser/{userid}/{roleid} API
4. Call /authenticate with username and password.
5. Copy the jwt token from response and append with **Bearer** and paste the full text into Authorization textbox and click login
6. Try out any endpoints under /product and verify the user is able to access only the end point he has access to in the roles.

**Points to be notes:**

1. Gateway app should have controllers for all microservices to which it interacts.
2. The data model used in microservices should also be used in the gateway.
3. Gateway will delegate the incoming request to appropriate microservices . Either spring cloud tech stack or kubernetes (TBD)