Oauth2

1. Implement an OAuth2 server
2. OAuth2 server should issue JWT
3. Make API Gateway as Resource Server
4. Enable API Gateway to validate JWT

Solution:

1. Oauth2 Server Implementation

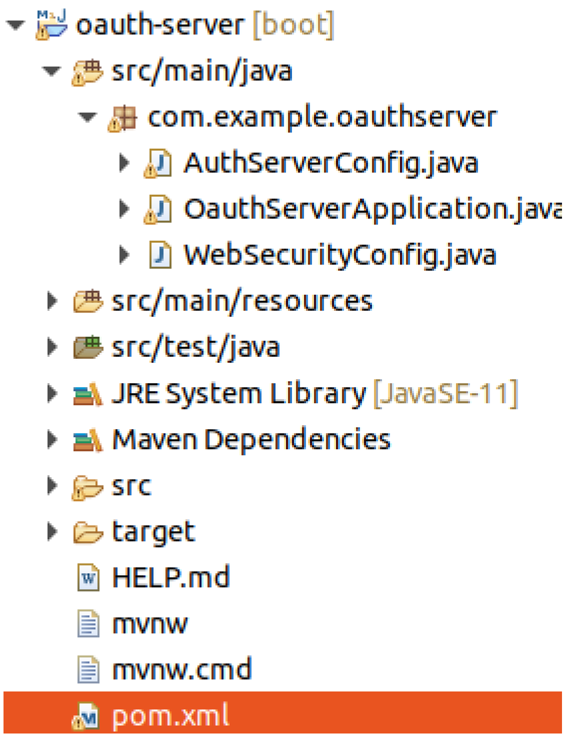
Dependendencies

- Spring web
- Eureka Client
- Oauth2

pom.xml

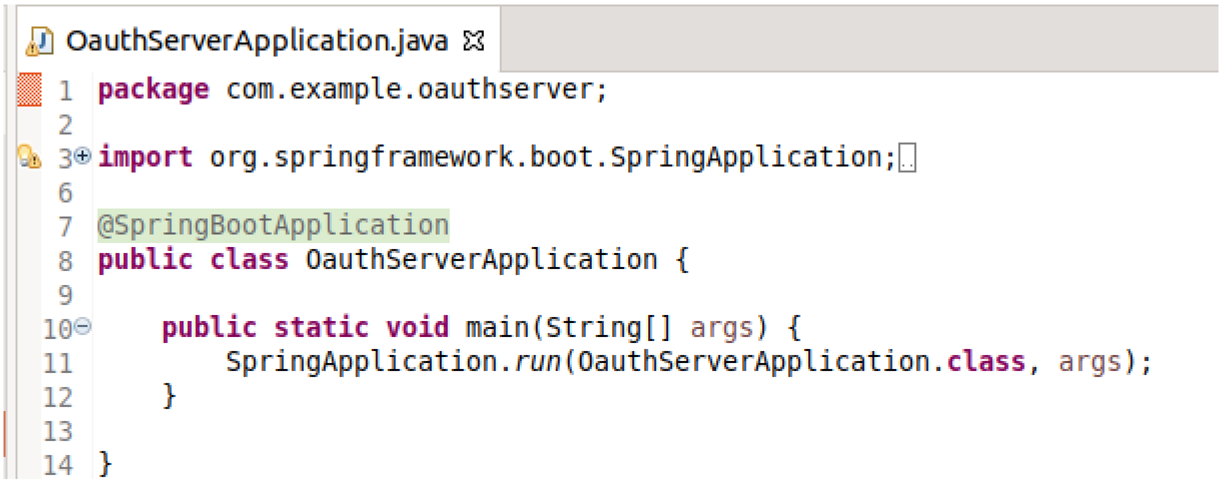
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
      <modelVersion>4.0.0</modelVersion>
      <parent>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-parent</artifactId>
            <version>2.6.4</version>
            <relativePath/> <!-- lookup parent from repository -->
      </parent>
      <groupId>com.example</groupId>
      <artifactId>oauth-server</artifactId>
      <version>0.0.1-SNAPSHOT</version>
      <name>oauth-server</name>
      <description>Demo project for Spring Boot</description>
      <properties>
            <java.version>11</java.version>
            <spring-cloud.version>2021.0.3</spring-cloud.version>
      </properties>
      <dependencies>
      <dependency>
    <groupId>org.glassfish.jaxb</groupId>
    <artifactId>jaxb-runtime</artifactId>
    <version>2.3.6</version>
</dependency>
            <dependency>
                  <groupId>org.springframework.cloud</groupId>
                  <artifactId>spring-cloud-starter-oauth2</artifactId>
                  <version>2.2.5.RELEASE</version>
            </dependency>
            <dependency>
                  <groupId>org.springframework.boot</groupId>
                  <artifactId>spring-boot-starter-web</artifactId>
            </dependency>
            <dependency>
                  <groupId>org.springframework.cloud</groupId>
                  <artifactId>spring-cloud-starter-netflix-eureka-
client</artifactId>
            </dependency>

            <dependency>
                  <groupId>org.springframework.boot</groupId>
                  <artifactId>spring-boot-starter-test</artifactId>
                  <scope>test</scope>
            </dependency>
      </dependencies>
      <dependencyManagement>
            <dependencies>
                  <dependency>
                        <groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-dependencies</artifactId>
                        <version>${spring-cloud.version}</version>
                        <type>pom</type>
                        <scope>import</scope>
                  </dependency>
            </dependencies>
      </dependencyManagement>

      <build>
            <plugins>
                  <plugin>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-maven-plugin</artifactId>
                  </plugin>
            </plugins>
      </build>

</project>
```

Project directory structure



```
oauth-server [boot]
  src/main/java
    com.example.oauthserver
      AuthServerConfig.java
      OauthServerApplication.java
      WebSecurityConfig.java
  src/main/resources
  src/test/java
  JRE System Library [JavaSE-11]
  Maven Dependencies
  src
  target
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
```

Main Application



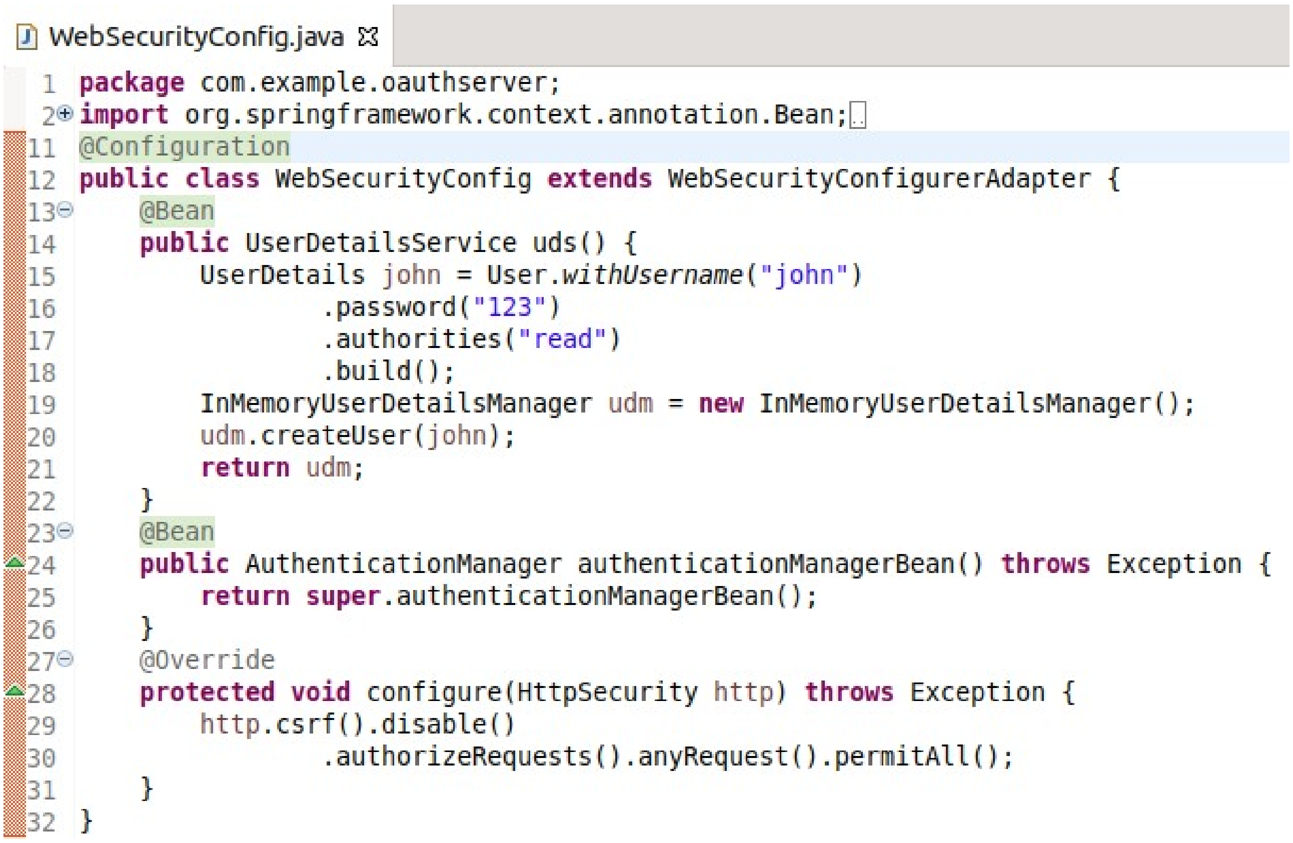
OauthServerApplication.java

```java
package com.example.oauthserver;

import org.springframework.boot.SpringApplication;

@SpringBootApplication
public class OauthServerApplication {

    public static void main(String[] args) {
        SpringApplication.run(OauthServerApplication.class, args);
    }

}
```

2. Issuing JWT from auth server

WebSecurityConfig and AuthServerConfig class for configiration of AuthServer to generate jwt containing access token and refresh token.

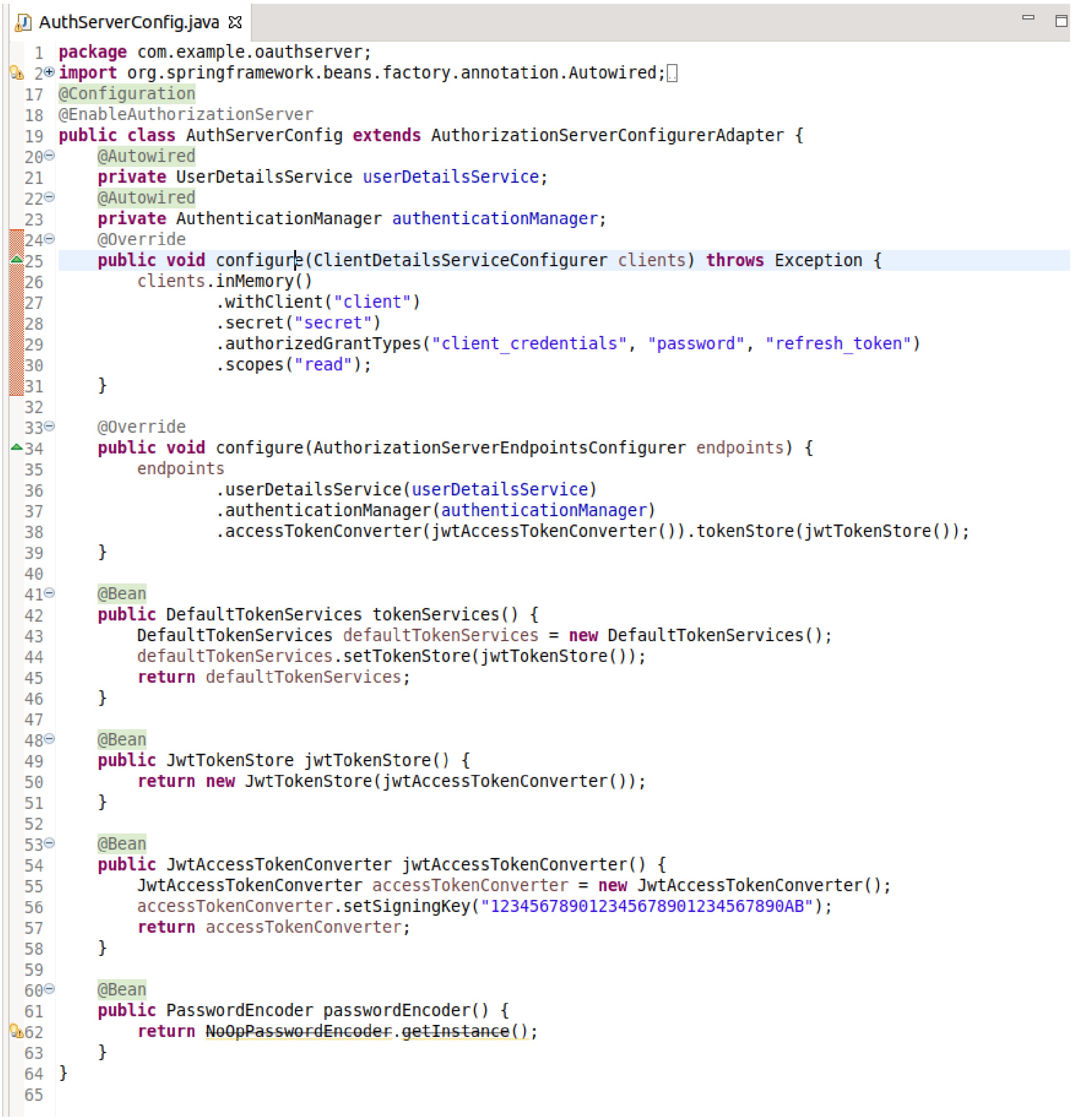
Creating in memoey user.



WebSecurityConfig.java

```java
package com.example.oauthserver;
import org.springframework.context.annotation.Bean;
@Configuration
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {
    @Bean
    public UserDetailsService uds() {
        UserDetails john = User.withUsername("john")
                .password("123")
                .authorities("read")
                .build();
        InMemoryUserDetailsManager udm = new InMemoryUserDetailsManager();
        udm.createUser(john);
        return udm;
    }
    @Bean
    public AuthenticationManager authenticationManagerBean() throws Exception {
        return super.authenticationManagerBean();
    }
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.csrf().disable()
                .authorizeRequests().anyRequest().permitAll();
    }
}
```

AuthServerConfig.class for generating jwt token



AuthServerConfig.java

```java
package com.example.oauthserver;
import org.springframework.beans.factory.annotation.Autowired;
@Configuration
@EnableAuthorizationServer
public class AuthServerConfig extends AuthorizationServerConfigurerAdapter {
    @Autowired
    private UserDetailsService userDetailsService;
    @Autowired
    private AuthenticationManager authenticationManager;
    @Override
    public void configure(ClientDetailsServiceConfigurer clients) throws Exception {
        clients.inMemory()
                .withClient("client")
                .secret("secret")
                .authorizedGrantTypes("client_credentials", "password", "refresh_token")
                .scopes("read");
    }

    @Override
    public void configure(AuthorizationServerEndpointsConfigurer endpoints) {
        endpoints
                .userDetailsService(userDetailsService)
                .authenticationManager(authenticationManager)
                .accessTokenConverter(jwtAccessTokenConverter()).tokenStore(jwtTokenStore());
    }

    @Bean
    public DefaultTokenServices tokenServices() {
        DefaultTokenServices defaultTokenServices = new DefaultTokenServices();
        defaultTokenServices.setTokenStore(jwtTokenStore());
        return defaultTokenServices;
    }

    @Bean
    public JwtTokenStore jwtTokenStore() {
        return new JwtTokenStore(jwtAccessTokenConverter());
    }

    @Bean
    public JwtAccessTokenConverter jwtAccessTokenConverter() {
        JwtAccessTokenConverter accessTokenConverter = new JwtAccessTokenConverter();
        accessTokenConverter.setSigningKey("123456789012345678901234567890AB");
        return accessTokenConverter;
    }

    @Bean
    public PasswordEncoder passwordEncoder() {
        return ~~NoOpPasswordEncoder.getInstance~~();
    }
}
```

3.Make API Gateway as Resource Server

Dependencies

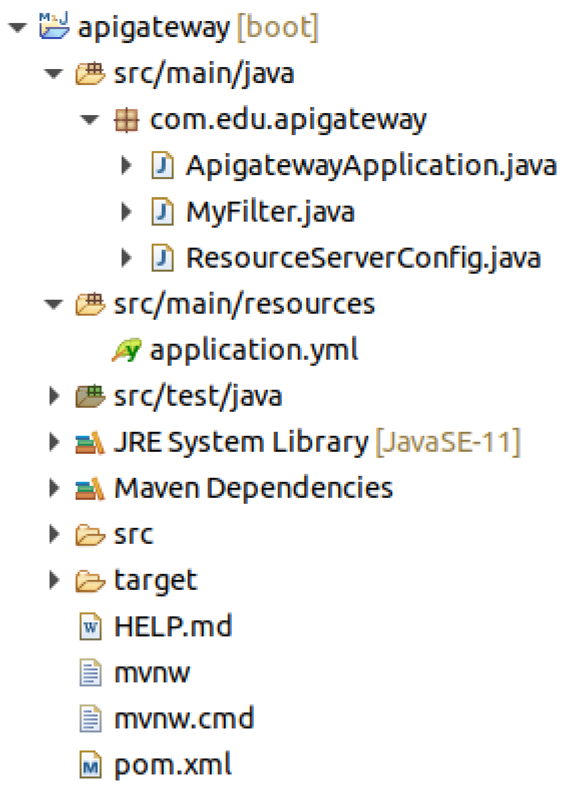
- **eureka-client**
- **cloud-gateway**
- **oauth2-resource-server**

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>2.7.2</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<groupId>com.edu</groupId>
	<artifactId>apigateway</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>apigateway</name>
	<description>Demo project for Spring Boot</description>
	<properties>
		<java.version>11</java.version>
		<spring-cloud.version>2021.0.3</spring-cloud.version>
	</properties>
	<dependencies>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-oauth2-resource-
server</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-actuator</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.cloud</groupId>
			<artifactId>spring-cloud-starter-gateway</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.cloud</groupId>
			<artifactId>spring-cloud-starter-netflix-eureka-
client</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.cloud</groupId>
			<artifactId>spring-cloud-starter-sleuth</artifactId>
		</dependency>

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

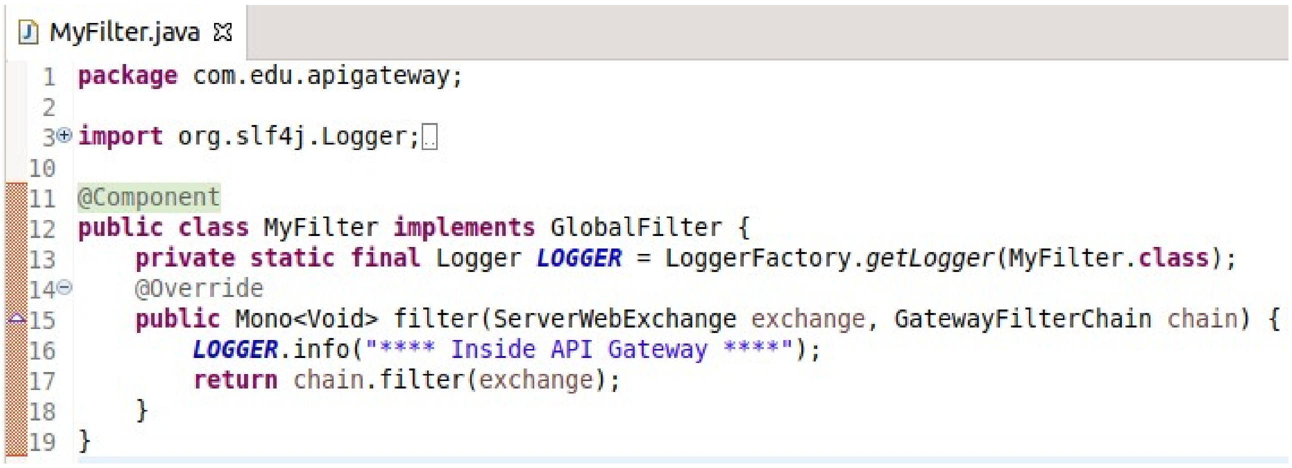
```
        </dependencies>
        <dependencyManagement>
                <dependencies>
                        <dependency>
                                <groupId>org.springframework.cloud</groupId>
                                <artifactId>spring-cloud-dependencies</artifactId>
                                <version>${spring-cloud.version}</version>
                                <type>pom</type>
                                <scope>import</scope>
                        </dependency>
                </dependencies>
        </dependencyManagement>

        <build>
                <plugins>
                        <plugin>
                                <groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-plugin</artifactId>
                        </plugin>
                </plugins>
        </build>

</project>
```

API Gateway Folder structer



```
apigateway [boot]
  src/main/java
    com.edu.apigateway
      ApigatewayApplication.java
      MyFilter.java
      ResourceServerConfig.java
  src/main/resources
    application.yml
  src/test/java
  JRE System Library [JavaSE-11]
  Maven Dependencies
  src
  target
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
```

Created MyFilter.java to filter all the request

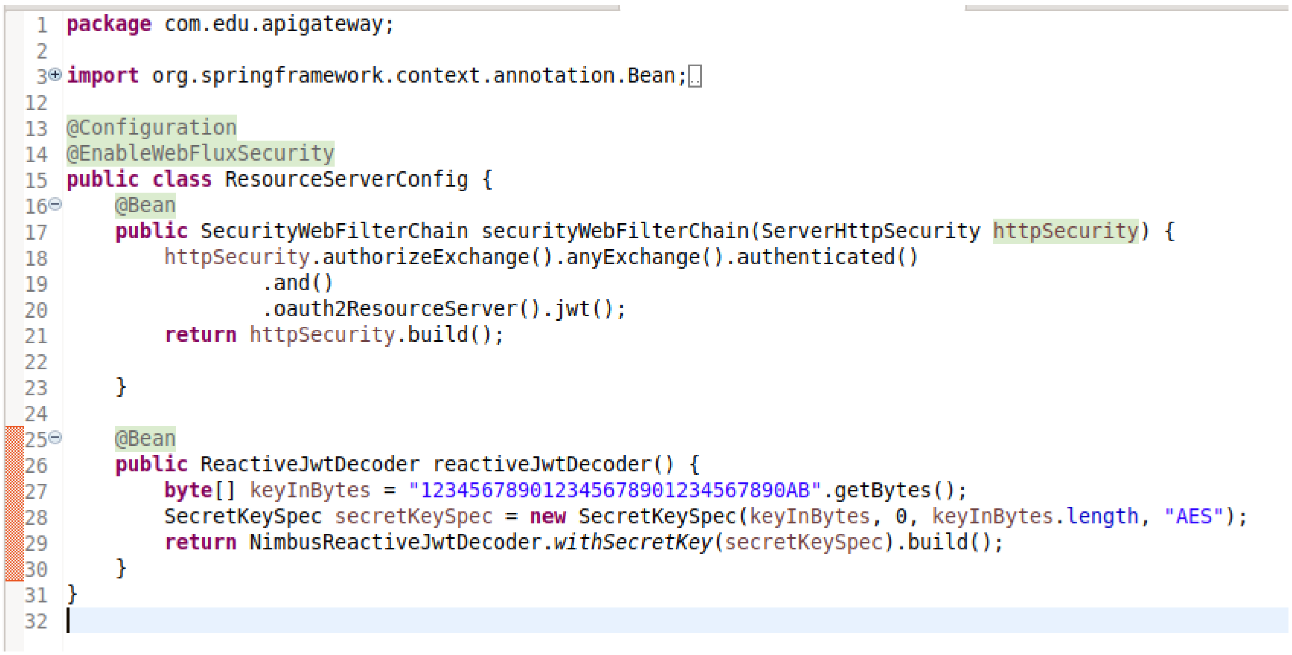


MyFilter.java

```java
package com.edu.apigateway;

import org.slf4j.Logger;

@Component
public class MyFilter implements GlobalFilter {
    private static final Logger LOGGER = LoggerFactory.getLogger(MyFilter.class);
    @Override
    public Mono<Void> filter(ServerWebExchange exchange, GatewayFilterChain chain) {
        LOGGER.info("**** Inside API Gateway ****");
        return chain.filter(exchange);
    }
}
```

4. Enable API Gateway to validate JWT

ResourceServerConfig.class



```java
package com.edu.apigateway;

import org.springframework.context.annotation.Bean;

@Configuration
@EnableWebFluxSecurity
public class ResourceServerConfig {
    @Bean
    public SecurityWebFilterChain securityWebFilterChain(ServerHttpSecurity httpSecurity) {
        httpSecurity.authorizeExchange().anyExchange().authenticated()
                .and()
                .oauth2ResourceServer().jwt();
        return httpSecurity.build();

    }

    @Bean
    public ReactiveJwtDecoder reactiveJwtDecoder() {
        byte[] keyInBytes = "12345678901234567890123456789OAB".getBytes();
        SecretKeySpec secretKeySpec = new SecretKeySpec(keyInBytes, 0, keyInBytes.length, "AES");
        return NimbusReactiveJwtDecoder.withSecretKey(secretKeySpec).build();
    }
}

```

Output:

Auth2 Server

terminal command to genereate jwt token in auth server

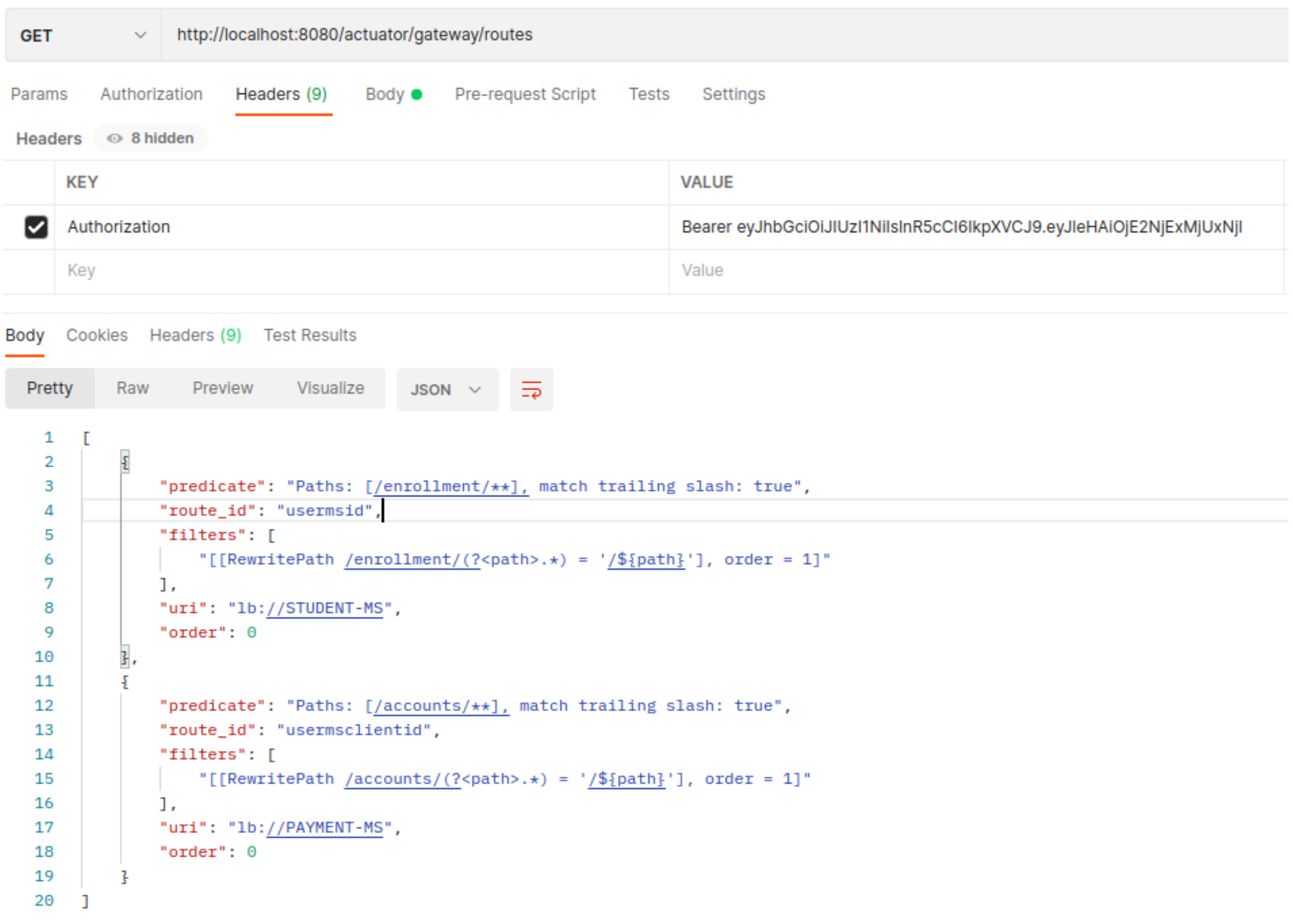
curl -v -X POST -u client:secret
'http://localhost:9000/oauth/tokengrant_type=password&username=john&password=123&scope=read'

{"access_token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2NjExMjY2NDcsInVzZXJfbmFtZSI6ImpvaG4iLCJhdXRob3JpdGllcyI6WyJyZWFkIl0sImp0aSI6IjUwZGU2MjUzLWFhNGUtNDRkZS05OGM1LTAxODU0NTJlYjQxZCIsImNsaWVudF9pZCI6ImNsaWVudCIsInNjb3BlIjpbInJlYWQiXX0.z7_HW_a8xRvXJnQIbEEMygfkdFJxI8rcAwUH6S54CIw","token_type":"bearer","refresh_token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX25hbWUiOiJqb2huIiwic2NvcGUiOlsicmVhZCJdLCJhdGkiOiI1MGRlNjI1My1hYTRlLTQ0ZGUtOThjNS0wMTg1NDUyZWI0MWQiLCJleHAiOjE2NjM2NzU0NDcsImF1dGhvcml0aWVzIjpbInJlYWQiXSwianRpIjoiNjczOWU0ZDUtMTQzOC00MmE3LWJhMWUtNzdlNjkzM2VmNTNhIiwiY2xpZW50X2lkIjoiY2xpZW50In0.dIgrmKrO6Wt_RC3X8KEZOmM63WR8vv9KteKxLV8fyA8","expires_in":43199,"scope":"read","jti":"50de6253-aa4e-44de-98c5-0185452eb41d"}

API Gateway

checking routes with token

http://localhost:8080/actuator/gateway/routes



GET http://localhost:8080/actuator/gateway/routes

Params | Authorization | Headers (9) | Body ● | Pre-request Script | Tests | Settings

Headers 8 hidden

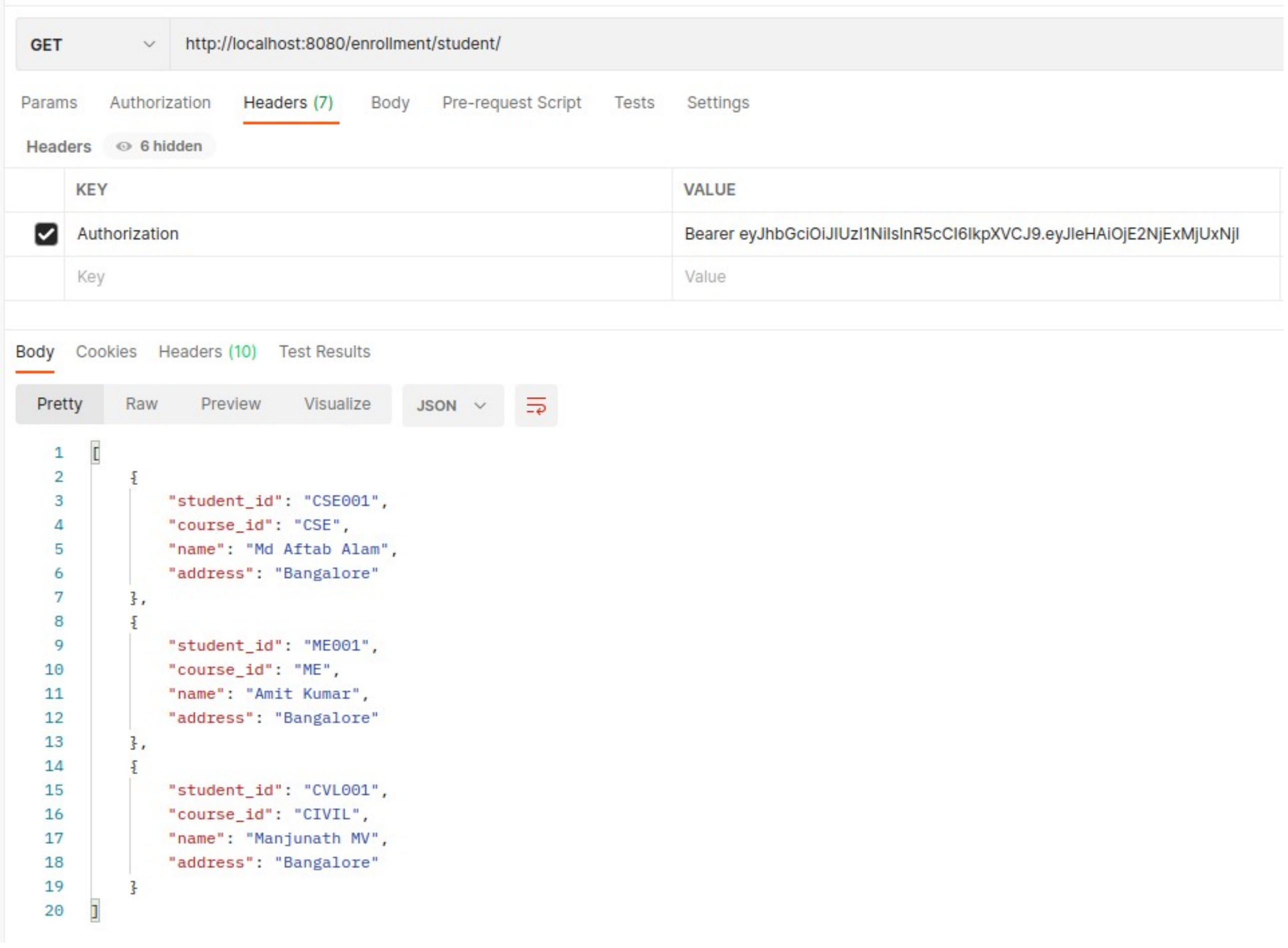
	KEY	VALUE
✓	Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2NjExMjUxNjI
	Key	Value

Body | Cookies | Headers (9) | Test Results

Pretty | Raw | Preview | Visualize | JSON

```
[
    {
        "predicate": "Paths: [/enrollment/**], match trailing slash: true",
        "route_id": "usermsid",
        "filters": [
            "[[RewritePath /enrollment/(?<path>.*) = '/${path}'], order = 1]"
        ],
        "uri": "lb://STUDENT-MS",
        "order": 0
    },
    {
        "predicate": "Paths: [/accounts/**], match trailing slash: true",
        "route_id": "usermsclientid",
        "filters": [
            "[[RewritePath /accounts/(?<path>.*) = '/${path}'], order = 1]"
        ],
        "uri": "lb://PAYMENT-MS",
        "order": 0
    }
]
```

Hitting routed API(http://localhost:8080/enrollment/student/) from API Gateway by providing Bearer token



GET http://localhost:8080/enrollment/student/

Headers (7)

KEY	VALUE
Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2NjExMjUxNjI

```
[
    {
        "student_id": "CSE001",
        "course_id": "CSE",
        "name": "Md Aftab Alam",
        "address": "Bangalore"
    },
    {
        "student_id": "ME001",
        "course_id": "ME",
        "name": "Amit Kumar",
        "address": "Bangalore"
    },
    {
        "student_id": "CVL001",
        "course_id": "CIVIL",
        "name": "Manjunath MV",
        "address": "Bangalore"
    }
]
```