**OAuth – Resource Server**

A diagram of a system

Description automatically generated

Get Auth Token

A screenshot of a computer

Description automatically generated

Get Access Token

Copy paste the previous pic url from postman to browser

http://localhost:8181/realms/appsdeveloperblog/protocol/openid-connect/auth?client\_id=photo-app-code-flow-client&response\_type=code&scope=openid profile&redirect\_uri=http://localhost:8083/callback&state=sdsdsdsdsdd

A screenshot of a computer

Description automatically generated

After login, get the auth code from the browser on the redirect\_uri provided

<http://localhost:8083/callback?state=sdsdsdsdsdd&session_state=892b8baa-758a-47f3-a952-de67a83aa7da&iss=http%3A%2F%2Flocalhost%3A8181%2Frealms%2Fappsdeveloperblog&code=11033b02-2d33-4f66-bfa0-bd774bc19a0a.892b8baa-758a-47f3-a952-de67a83aa7da.9fafc20b-9b30-4dee-88aa-da3ee83b956d>

A screenshot of a computer

Description automatically generated

User the code, and use postman with POST call , get the access token in the response

A screenshot of a computer

Description automatically generated

User access token, to call the api – thru postman as below

A screenshot of a computer

Description automatically generated

The Source code as below

**pom.xml**

<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-web</artifactId>  
</dependency>

**application.properties**

spring.application.name=ResourceServer  
server.port=8081  
spring.security.oauth2.resourceserver.jwt.jwk-set-uri=http://localhost:8181/realms/appsdeveloperblog/protocol/openid-connect/certs

@RestController  
@RequestMapping("/token")  
public class TokenController {  
  
 @GetMapping  
 public Map<String, Object> getToken(@AuthenticationPrincipal Jwt jwt) {  
 return Collections.*singletonMap*("principal",jwt);  
 }  
}

**Scope Based Resource Access**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Offline\_access scope is used to get refresh\_token

**Userinfo**

**A screenshot of a computer

Description automatically generated**

**Valid scope (200) & Invalid scope (403)**

A screenshot of a computer

Description automatically generated

**Role Based Authorities**

**A screenshot of a software application

Description automatically generated**

**A white paper with black text and arrows

Description automatically generated**

Create realm level role – “developer”

A screenshot of a computer

Description automatically generated

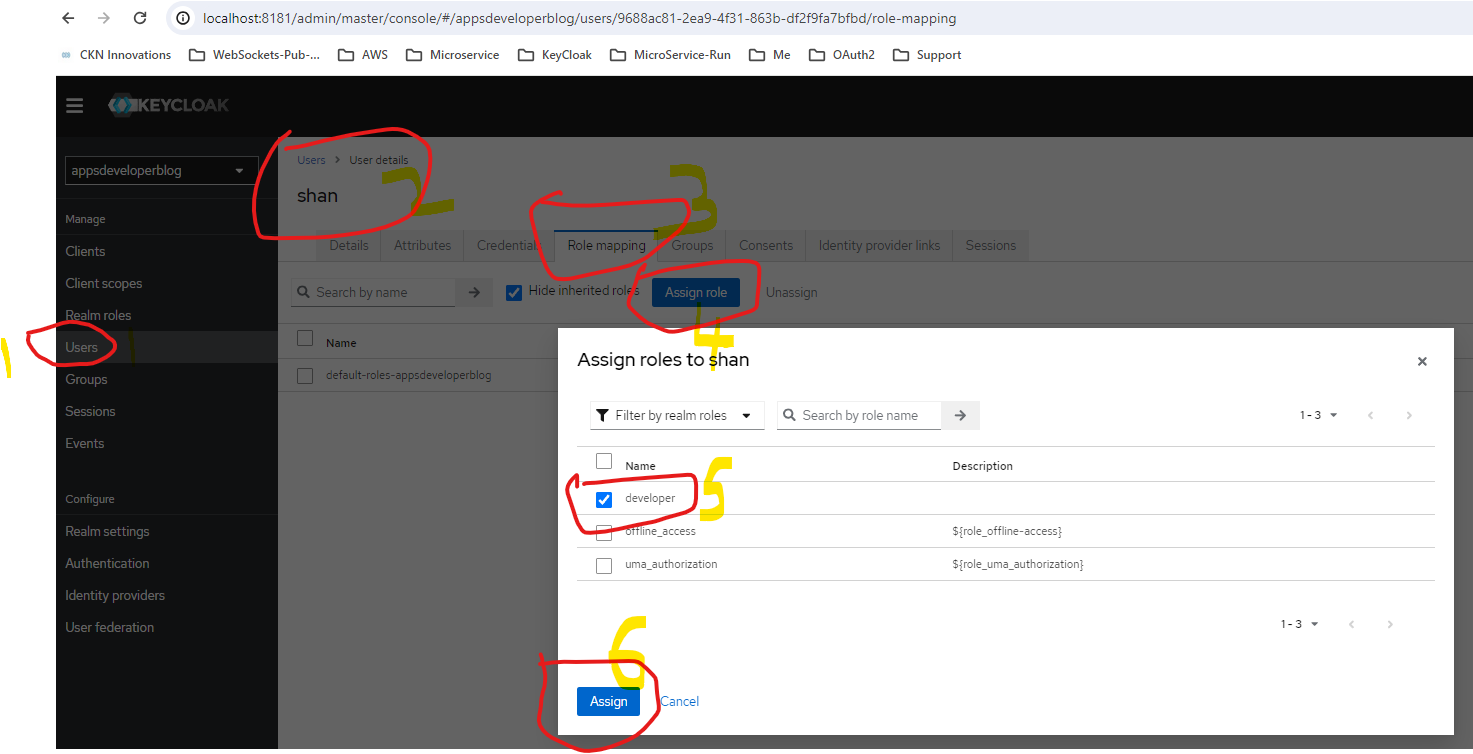
Add user to the Role “Developer”

Go to Users menu

A screenshot of a computer

Description automatically generated

Select user and assign previously created “Developer” role (from realm level)



Get auth code by logging in

http://localhost:8181/realms/appsdeveloperblog/protocol/openid-connect/auth?client\_id=photo-app-code-flow-client&response\_type=code&scope=openid profile&redirect\_uri=http://localhost:8083/callback&state=sdsdsdsdsdd

From browser Login, and get the “code”

http://localhost:8083/callback?state=sdsdsdsdsdd&session\_state=d8530179-9657-426d-a9cd-3471898d521a&iss=http%3A%2F%2Flocalhost%3A8181%2Frealms%2Fappsdeveloperblog&code=167f29d3-eaa6-42f7-98ff-802ac396d63b.d8530179-9657-426d-a9cd-3471898d521a.9fafc20b-9b30-4dee-88aa-da3ee83b956d

Get Authorization code by POST call

A screenshot of a computer

Description automatically generated

Check JWT token from jwt.io, to get the roles array (check ‘developer’ role , created earlier from key cloak is there)

A screenshot of a computer

Description automatically generated

Code to convert jwt roles to Collection<GrantedAuthority>

public class KeycloakRoleConverter implements Converter<Jwt, Collection<GrantedAuthority>> {  
 @Override  
 public Collection<GrantedAuthority> convert(Jwt jwt) {  
 Map<String, Object> realmAccess = (Map<String, Object>) jwt.getClaims().get("realm\_access");  
 if (realmAccess == null || realmAccess.isEmpty()) {  
 return new ArrayList<>();  
 }  
 List<String> roles = (List<String>) realmAccess.get("roles");  
 Collection<GrantedAuthority> returnValue = roles.stream()  
 .map(roleName -> "ROLE\_"+roleName)  
 .map(SimpleGrantedAuthority::new)  
 .collect(Collectors.*toList*());  
 return returnValue;  
 }  
}

**Method Level Security**

A close-up of a message

Description automatically generated

@Configuration  
@EnableWebSecurity  
@EnableMethodSecurity(securedEnabled=true, prePostEnabled=true)  
public class SpringSecurity {  
 @Bean  
 public SecurityFilterChain configure(HttpSecurity http) throws Exception {  
 JwtAuthenticationConverter jwtAuthenticationConverter = new JwtAuthenticationConverter();  
 jwtAuthenticationConverter.setJwtGrantedAuthoritiesConverter(new KeycloakRoleConverter());  
 http.authorizeHttpRequests(auth ->  
 auth  
 .requestMatchers(HttpMethod.*GET*, "/users/status/check")  
.hasRole("developer")  
.anyRequest().authenticated())  
 .oauth2ResourceServer(oauth2 -> oauth2.jwt( jwt ->{  
 jwt.jwtAuthenticationConverter(jwtAuthenticationConverter);  
 }));  
 return http.build();  
 }  
}

@Configuration  
@EnableWebSecurity  
@EnableMethodSecurity(securedEnabled=true, **prePostEnabled=true**)  
public class SpringSecurity {

A close-up of a computer

Description automatically generated

@Configuration  
@EnableWebSecurity  
@EnableMethodSecurity(securedEnabled=true, prePostEnabled=true)  
public class SpringSecurity {  
 @Bean  
 public SecurityFilterChain configure(HttpSecurity http) throws Exception {  
 JwtAuthenticationConverter jwtAuthenticationConverter = new JwtAuthenticationConverter();  
 jwtAuthenticationConverter.setJwtGrantedAuthoritiesConverter(new KeycloakRoleConverter());  
 http.authorizeHttpRequests(auth ->  
 auth  
 .requestMatchers(HttpMethod.*GET*, "/users/status/check")  
.hasRole("developer").anyRequest().authenticated())  
 .oauth2ResourceServer(oauth2 -> oauth2.jwt( jwt ->{  
jwt.jwtAuthenticationConverter(jwtAuthenticationConverter);  
 }));  
 return http.build();  
 }  
}

A screenshot of a computer code

Description automatically generated@PostAuthorize("returnObject.id == #jwt.subject")  
@GetMapping(path = "/{id}")  
public UserRest getUser(@PathVariable String id,  
 @AuthenticationPrincipal Jwt jwt) {  
 return new UserRest("9688ac81-2ea9-4f31-863b-df2f9fa7bfbd", "Shanmugham", "Sundaram");  
}

**Resource Server behind API Gateway**

**A diagram of a software application

Description automatically generated**

A screenshot of a computer screen

Description automatically generated

Password Grant – POST Call

A screenshot of a computer

Description automatically generated

Spring MVC Application with Auth Server /Gateway/

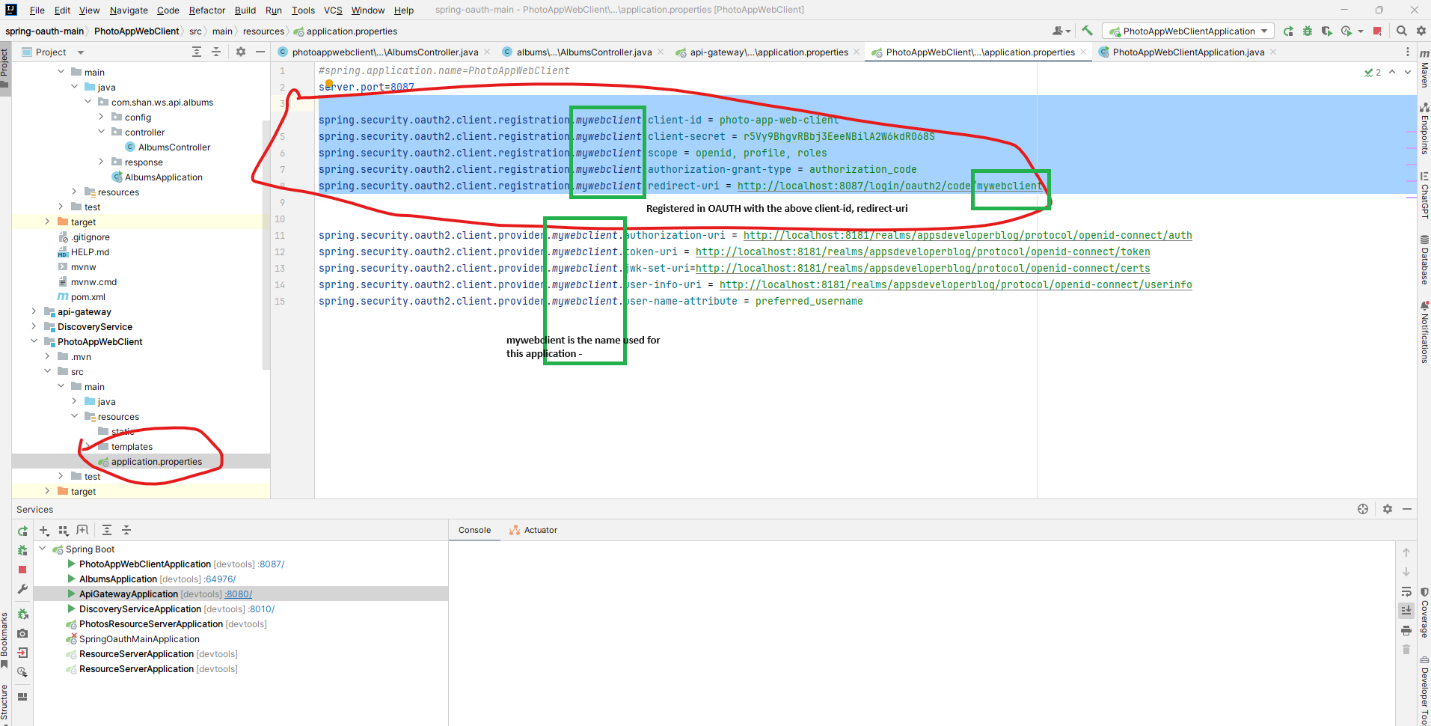
A diagram of a computer program

Description automatically generated

Enable client-id, secret for our web app, create new client with below setttings

A screenshot of a computer

Description automatically generated

OAuth client params, and access end points – in Application properties for Spring MVC client

**OAUTH2 – Social Login**

A screenshot of a computer

Description automatically generated

Disable PKCE

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated