

Rudraksh Chourey Assignment 4

Google oath client creation

The screenshot shows the Google Cloud Platform console for a project named 'spring-oauth-demo'. The 'Clients' page is active, displaying the configuration for a 'Client ID for Web application'.

Client ID for Web application

Name: spring-test-client

Additional information:

- Client ID:** 597549050765-ta00tupm62a3tp6d66mmhuhlr.apps.googleusercontent.com
- Creation date:** October 9, 2025 at 5:17:25 PM GMT+5
- Last used date:** October 9, 2025 (Note: this data could be delayed by a day or more.)

Authorized JavaScript origins:

- URI 1:** https://www.example.com

Authorized redirect URIs:

- URI 1:** http://localhost:8080/login/oauth2/code/google

Client secrets:

- Client secret:** ****WC4
- Creation date:** October 9, 2025 at 5:17:25 PM GMT+5
- Status:** Enabled

Security Config:

```
package com.assignment4.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.web.SecurityFilterChain;

@Configuration
@EnableWebSecurity
public class SecurityConfig {

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        http
            .authorizeHttpRequests(auth -> auth
                .requestMatchers("/hello").authenticated()
                .anyRequest().permitAll()
            )
            .oauth2Login(); // Enables OAuth2 login with configured providers
        return http.build();
    }
}
```

Controller:

```
package com.assignment4.controller;

import org.springframework.security.oauth2.core.user.OAuth2User;
import org.springframework.web.servlet.mvc.annotation.annotation.AnnotationMethodHandlerMethod;
import org.springframework.web.servlet.mvc.annotation.annotation.AnnotationMethodHandlerMethod;

@RestController
public class HelloController {

    @GetMapping("/hello")
    public String hello(Principal principal) {
        if (principal instanceof OAuth2User oAuth2User) {
            String name = oAuth2User.getAttribute("name");
            return "Hello, " + name + "! You logged in via OAuth2 ";
        }
        return "Hello, " + principal.getName();
    }
}
```

Login Page:

The image shows two browser windows. The top window is a Google login page with the URL `accounts.google.com/v3/signin/identifier?opparams=%253F&dsh=S-1002796841%3A1760080055273361&client_id=597549050765-tal0...`. It features the 'Sign in with Google' header, a 'Sign in' section with the text 'to continue to Spring Boor OAuth Demo', an input field for 'Email or phone', a 'Forgot email?' link, and buttons for 'Create account' and 'Next'. The bottom window is a local development page with the URL `localhost:8080/hello?continue`. It displays the text 'Hello, 110050032130039120461'.

Sign in with Google

Sign in

to continue to [Spring Boor OAuth Demo](#)

Email or phone

[Forgot email?](#)

[Create account](#) [Next](#)

English (United States) [Help](#) [Privacy](#) [Terms](#)

localhost:8080/hello?continue

localhost:8080/hello?continue

Hello, 110050032130039120461

Swagger First Exercise:

Swagger Config:

```

HelloController.java  SwaggerConfig.java X
1 package com.assignment4b.config;
2
3+ import io.swagger.v3.oas.models.OpenAPI;
7 @Configuration
8 public class SwaggerConfig {
9     @Bean
10    public OpenAPI customOpenAPI() {
11        return new OpenAPI()
12            .info(new Info()
13                .title("My School API")
14                .version("1.0")
15                .description("API documentation for my Spring Boot project"));
16    }
17 }

```

Controller:

```

HelloController.java X  SwaggerConfig.java
1 package com.assignment4b.config;
2
3- import org.springframework.web.bind.annotation.*;
4
5 import io.swagger.v3.oas.annotations.Operation;
6 import io.swagger.v3.oas.annotations.responses.ApiResponse;
7 @RestController
8 @RequestMapping("/api/hello")
9 public class HelloController {
10     @GetMapping
11     @Operation(summary = "Say Hello", description = "Returns a greeting message")
12     @ApiResponse(responseCode = "200", description = "Successful greeting")
13-    public String sayHello() {
14        return "Hello, Swagger!";
15    }
16    @GetMapping("/{name}")
17-    public String greetByName(@PathVariable String name) {
18        return "Hello, " + name + "!";
19    }
20 }
21

```

Swagger first ex 1:

First mapping:

hello-controller

GET /api/hello Say Hello

Returns a greeting message

Parameters Cancel

No parameters

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8080/api/hello' \
  -H 'accept: */*'
```

Request URL

http://localhost:8080/api/hello

Server response

Code	Details
200	<p>Response body</p> <p>Hello, Swagger!</p> <p>Download</p> <p>Response headers</p> <p>connection: keep-alive content-length: 15 content-type: text/plain; charset=UTF-8 date: Fri, 18 Oct 2025 12:35:45 GMT keep-alive: timeout=60</p>

Response

Code	Description	Links
200	<p>Successful greeting</p> <p>Media type</p> <p>application/json</p> <p>Content Accept header</p> <p>Example Value Schema</p> <p>string</p>	No links

Second Mapping:

GET

/api/hello/{name}

Parameters

Cancel

Name	Description
name	
string	Rudraksh
(path)	

Execute

Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8080/api/hello/Rudraksh' \
  -H 'accept: */*'
```

Request URL

http://localhost:8080/api/hello/Rudraksh

Server response

Code	Details
200	<div>Response body</div> <div>Hello, Rudraksh!</div> <div>Download</div> <div>Response headers</div> <div>connection: keep-alive content-length: 16 content-type: text/plain; charset=utf-8 date: Fri, 18 Oct 2025 12:14:03 GMT keep-alive: timeout=60</div>

Response

Code	Description	Links
200	OK	No links

Media type

/

Content Accept Header

Example Value | Schema

string

OpenAPI JSON:

```
{
  "openapi": "3.0.1",
  "info": {
    "title": "My School API",
    "description": "API documentation for my Spring Boot project",
    "version": "1.0"
  },
  "servers": [
    {
      "url": "http://localhost:8080",
      "description": "Generated server url"
    }
  ],
  "paths": {
    "/api/hello": {
      "get": {
        "tags": [
          "hello-controller"
        ],
        "summary": "Say Hello",
        "description": "Returns a greeting message",
        "operationId": "sayHello",
        "responses": {
          "200": {
            "description": "Successful greeting",
            "content": {
              "*/*": {
                "schema": {
                  "type": "string"
                }
              }
            }
          }
        }
      }
    }
  }
},
```

```
"/api/hello/{name}": {
  "get": {
    "tags": [
      "hello-controller"
    ],
    "operationId": "greetByName",
    "parameters": [
      {
        "name": "name",
        "in": "path",
        "required": true,
        "schema": {
          "type": "string"
        }
      }
    ],
    "responses": {
      "200": {
        "description": "OK",
        "content": {
          "*/*": {
            "schema": {
              "type": "string"
            }
          }
        }
      }
    }
  }
},
"components": {
}
```


Security Config Code:

```
SecurityConfig.java X
1 package com.assignment4b.config;
2
3 import org.springframework.context.annotation.Bean;
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21 @Configuration
22 @EnableWebSecurity
23 public class SecurityConfig {
24
25     private final AuthController
26     authController;
27
28     public SecurityConfig(AuthController authController) {
29         this.authController = authController;
30     }
31
32     @Bean
33     public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
34         http.csrf(csrf -> csrf.disable()) // Disable CSRF for Stateless API (JWT)
35
36         // 1. Authorization Rules (Order Matters: PermitAll first!)
37         .authorizeHttpRequests(auth -> auth
38             // Allow public access for Swagger UI, API docs, and JWT login/token generation
39             .requestMatchers("/swagger-ui/**", "/v3/api-docs/**", "/api/auth/**").permitAll()
40
41             // Require authentication for /hello to trigger the OAuth2 login flow
42             .requestMatchers("/hello").authenticated()
43
44             // Require ADMIN role for admin endpoints (JWT-protected)
45             .requestMatchers("/api/admin/**").hasRole("ADMIN")
46
47             // All other requests require authentication (will be caught by either OAuth2 Login or JWT)
48             .anyRequest().authenticated()
49         )
50
51         // 2. Enable Session-based Login (Google OAuth2 Client)
52         // This is primarily used for browser access to endpoints like /hello
53         .oauth2Login();
54
55         // 3. Enable Token-based Authentication (JWT Resource Server)
56         // This is primarily used for API calls to endpoints like /api/secure
57         http.oauth2ResourceServer(oauth2 -> oauth2.jwt(jwt ->
58             jwt.jwtAuthenticationConverter(this::convertJwt)));
59
60         return http.build();
61     }
62
63     /**
64      * Configures the JwtDecoder to use the secret key generated by the AuthController
65      */
66     @Bean
67     public JwtDecoder jwtDecoder() {
68         return NimbusJwtDecoder.withSecretKey((SecretKey) authController.getKey()).build();
69     }
70
71     /**
72      * Customizes the JWT authentication process to extract the 'roles' claim
73      * and map them to Spring Security authorities.
74      */
75     private AbstractAuthenticationToken convertJwt(Jwt jwt) {
76         // Extracts 'roles' claim (which is a List<String>) and maps them to SimpleGrantedAuthority
77         Collection<GrantedAuthority> authorities = ((List<String>)
78             jwt.getClaims().get("roles")).stream()
79             .map(SimpleGrantedAuthority::new)
80             .collect(Collectors.toList());
81
82         return new JwtAuthenticationToken(jwt, authorities);
83     }
84 }
```

Activate Window

Activate Window

Auth Controller:

```
SecurityConfig.java  AuthController.java X
1 package com.assignment4b.controller;
2
3 import io.jsonwebtoken.Jwts;
4 import io.jsonwebtoken.SignatureAlgorithm;
5 import io.jsonwebtoken.security.Keys;
6 import org.springframework.web.bind.annotation.*;
7 import java.security.Key;
8 import java.util.Date;
9 import java.util.List;
10
11 @RestController
12 @RequestMapping("/api/auth")
13 public class AuthController {
14
15     private final Key key = Keys.secretKeyFor(SignatureAlgorithm.HS256);
16
17     @PostMapping("/login")
18     public String login(@RequestParam String username,
19                       @RequestParam(defaultValue = "ROLE_USER") String role) {
20         return Jwts.builder()
21             .setSubject(username)
22             .claim("roles", List.of(role)) // embedding roles into token
23             .setIssuedAt(new Date())
24             .setExpiration(new Date(System.currentTimeMillis() + 3600000)) // 1 hour expiry
25             .signWith(key)
26             .compact();
27     }
28
29     public Key getKey() {
30         return key;
31     }
32 }
```

Secure Controller:

```
SecurityConfig.java  AuthController.java  SecureController.java X
1 package com.assignment4b.controller;
2
3 import org.springframework.web.bind.annotation.*;
4 @RestController
5 @RequestMapping("/api")
6 public class SecureController {
7     @GetMapping("/secure")
8     public String userEndpoint() {
9         return "Hello USER - you are authenticated!";
10    }
11    @GetMapping("/admin/secure")
12    public String adminEndpoint() {
13        return "Hello ADMIN - you have admin privileges!";
14    }
15 }
16 }
```


\wedge

GET

/api/admin/secure

Parameters

Cancel

No parameters

Execute

Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8080/api/admin/secure' \
  -H 'accept: */*' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXLTJ5IiwiaWF0IjoxNjU0MzUwMDAwLCJ0eXAiOiJKV1QiLCJhdGUiOiJ1bm90eSIsImF1dG8iOiJ1bm90eSIsImV4cCI6MTY1NDQ1MDAwfQ.eyJ0eXAiOiJKV1QiLCJhdGUiOiJ1bm90eSIsImF1dG8iOiJ1bm90eSIsImV4cCI6MTY1NDQ1MDAwfQ.eyJ0eXAiOiJKV1QiLCJhdGUiOiJ1bm90eSIsImF1dG8iOiJ1bm90eSIsImV4cCI6MTY1NDQ1MDAwfQ'
```

Request URL

http://localhost:8080/api/admin/secure

Server response

Code	Details
200	<div><div>Response body</div><div>Hello ADMIN - you have admin privileges!</div><div> Download</div></div> <div><div>Response headers</div><div>cache-control: no-cache, no-store, max-age=0, must-revalidate connection: keep-alive content-length: 40 content-type: text/plain; charset=UTF-8 date: Fri, 10 Oct 2025 14:07:41 GMT expires: 0 keep-alive: timeout=60 pragma: no-cache x-content-type-options: nosniff x-frame-options: DENY x-oxp-protection: 0</div></div>

Responses

Code	Description	Links
200	OK	No links

Media type

/

Controls Accept header

Example Value | Schema

string