**OAuth2 in Spring Boot 3.4 using Google**:

**Step 1: Set Up Google OAuth2 Credentials**

1. **Create a Google Cloud Project**:
   * Visit the Google Cloud Console.
   * Create or select a project.
2. **Enable the "OAuth2.0 API"**:
   * Go to **APIs & Services > Library**.
   * Enable the **"OAuth2.0 API"** for your project.
3. **Create OAuth 2.0 Credentials**:
   * Go to **APIs & Services > Credentials**.
   * Click **Create Credentials > OAuth 2.0 Client IDs**.
   * Choose **Web application** as the application type.
   * Set the **Authorized redirect URIs**:
     + Example: http://localhost:8080/login/oauth2/code/google.
   * Copy the Client ID and Client Secret.

**Step 2: Create a Spring Boot Project**

1. Use [Spring Initializr](https://start.spring.io/):
   * **Dependencies**:
     + Spring Web
     + Spring Security
     + OAuth2 Client
   * **Spring Boot Version**: 3.4.x
2. Download the project and unzip it.

**Step 3: Configure application.yml**

Add your Google OAuth2 credentials and configure OAuth2 settings in src/main/resources/application.yml:

spring:

security:

oauth2:

client:

registration:

google:

client-id: YOUR\_CLIENT\_ID

client-secret: YOUR\_CLIENT\_SECRET

scope:

- profile

- email

server:

port: 10001

Replace YOUR\_CLIENT\_ID and YOUR\_CLIENT\_SECRET with the values from Google.

**Step 4: Security Configuration**

Create a configuration class to secure your application and enable OAuth2 login.

**SecurityConfig.java**

package com.mybanking.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.web.SecurityFilterChain;

@Configuration

public class SecurityConfig {

@Bean

public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

http

.authorizeHttpRequests(authorize -> authorize

.requestMatchers("/", "/login", "/oauth2/\*\*").permitAll()

.anyRequest().authenticated()

)

.oauth2Login();

return http.build();

}

}

**Step 5: Add a Controller for User Information**

Create a controller to display the authenticated user's information.

**EmployeeController.java**

package com.example.oauth2google.controller;

import org.springframework.security.core.annotation.AuthenticationPrincipal;

import org.springframework.security.oauth2.core.user.OAuth2User;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

import java.util.Map;

@RestController

public class EmployeeController {

@GetMapping("/user")

public Map<String, Object> userInfo(@AuthenticationPrincipal OAuth2User principal) {

return principal.getAttributes();

}

}

**Step 6: Run the Application**

1. Start your Spring Boot application:

mvn spring-boot:run

1. Visit the application:
   * Access http://localhost:10001/listemp.
   * Click **Login with Google**.

**Step 7: Test the OAuth2 Integration**

1. Authenticate with Google.
2. After logging in, you'll be redirected to / or the default login success page.
3. Access /emplist to view user details (e.g., email, name).

**Step 8: Customize OAuth2 Login**

To add a custom login page:

1. Create an HTML login page in src/main/resources/templates:

**login.html**

<!DOCTYPE html>

<html>

<head>

<title>Login</title>

</head>

<body>

<h1>Login</h1>

<a href="/oauth2/authorization/google">Login with Google</a>

</body>

</html>

1. Update SecurityConfig to specify the login page:

@Bean

public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

http

.authorizeHttpRequests(authorize -> authorize

.requestMatchers("/", "/login", "/oauth2/\*\*").permitAll()

.anyRequest().authenticated()

)

.oauth2Login(oauth2 -> oauth2

.loginPage("/login")

);

return http.build();

}