Spring Boot OAuth2.0 Security

1. Add dependencies
2. Configure Spring Security for OAuth2.
3. Set up Resource Server configuration
4. Test the OAuth2 security

Implementation Steps

1. Add the OAuth2 dependencies

Pom.xml

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-oauth2-client</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-oauth2-resource-server</artifactId>

</dependency>

1. Configure Spring Security for OAuth2.

**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(authorizeRequests->

authorizeRequests

.requestMatchers("/books").authenticated()

.anyRequest().permitAll()

).oauth2Login()

.and()

.logout()

.logoutUrl("/")

.permitAll();

**return** http.build();

}

}

1. Set up Resource Server configuration:-

set up **Google OAuth2** as a resource server and get the credentials required for authentication (like client\_id and client\_secret), follow these steps:

**1. Create a Google Cloud Project**

1. Go to the Google Cloud Console.
2. Sign in with your Google account if you haven’t already.
3. Click the **Select a project** dropdown at the top and choose **New Project**.
4. Enter a **Project Name** and click **Create**.

**2. Enable Google OAuth2 API**

1. In the **Google Cloud Console**, click on the **Navigation Menu** (three horizontal lines) in the top-left corner.
2. Go to **APIs & Services** → **Library**.
3. In the search bar, type **Google+ API** (for Google sign-in functionality) or **Google Identity Services** for a more comprehensive OAuth experience.
4. Click on **Google+ API** or **Google Identity Services** from the search results and then click **Enable**.

**3. Create OAuth2 Credentials**

1. In the **Google Cloud Console**, go to **APIs & Services** → **Credentials**.
2. Click on **+ CREATE CREDENTIALS** and select **OAuth 2.0 Client IDs**.
3. You’ll need to configure your OAuth consent screen first.

**OAuth Consent Screen** setup:

* + Choose **External** (for apps that will be used by any Google account) or **Internal** (for G Suite/Workspace users within an organization).
  + Fill in the required details like **App name**, **Support email**, and **Developer contact info**.
  + Add **Authorized domains** (e.g., localhost for testing or your actual domain).
  + Click **Save and Continue**.

1. Once the consent screen is set, return to **Credentials** and continue creating **OAuth 2.0 Client IDs**.
2. Choose **Web application** as the **Application type**.
3. In the **Authorized redirect URIs**, add the following URIs (depending on your environment):
   * For local development: http://localhost:8082/login/oauth2/code/google
   * For production: Add your production redirect URI (e.g., https://yourdomain.com/login/oauth2/code/google).
4. Click **Create**.

**4. Get OAuth2 Client ID and Secret**

After successfully creating the OAuth credentials, Google will provide:

* **Client ID**
* **Client Secret**

Download these credentials or copy them for use in your Spring Boot configuration.

**5. Configure OAuth2 Consent Screen (Optional)**

1. If your application is in development, the consent screen will display a warning for users as an **Unverified app**. You can ignore this for testing purposes.
2. For production apps, you must go through the **OAuth verification** process to remove the warning and increase the user limit.

**6. Add Authorized Scopes**

When configuring the **OAuth2 consent screen**, you’ll be asked to specify the scopes your app will request. Some common scopes are:

* email
* profile
* openid

You can choose these depending on the data you want to access from the user.

**7. Use OAuth2 Client ID and Secret in Spring Boot**

Now that you have the **client\_id** and **client\_secret**, use them in your Spring Boot project for OAuth2 integration. Add them to the **application.properties** or **application.yml** file as described previously.

For example:

properties

Copy code

spring.security.oauth2.client.registration.google.client-id=YOUR\_GOOGLE\_CLIENT\_ID

spring.security.oauth2.client.registration.google.client-secret=YOUR\_GOOGLE\_CLIENT\_SECRET

spring.security.oauth2.client.registration.google.scope=openid,profile,email

spring.security.oauth2.client.registration.google.redirect-uri={baseUrl}/login/oauth2/code/{registrationId}

**4. Test the OAuth Flow**

* When you run your Spring Boot application, it should redirect you to Google’s OAuth2 login page when you try to access a secured endpoint.
* After successfully logging in with your Google account, you’ll be redirected back to your application.

**Microservices Communication**

**BorrowService -> BookService**

**RestTemplate -> bean-> Spring Boot Framework**

**OrderService -> EmailService**