

14.Google OAuth2 Login

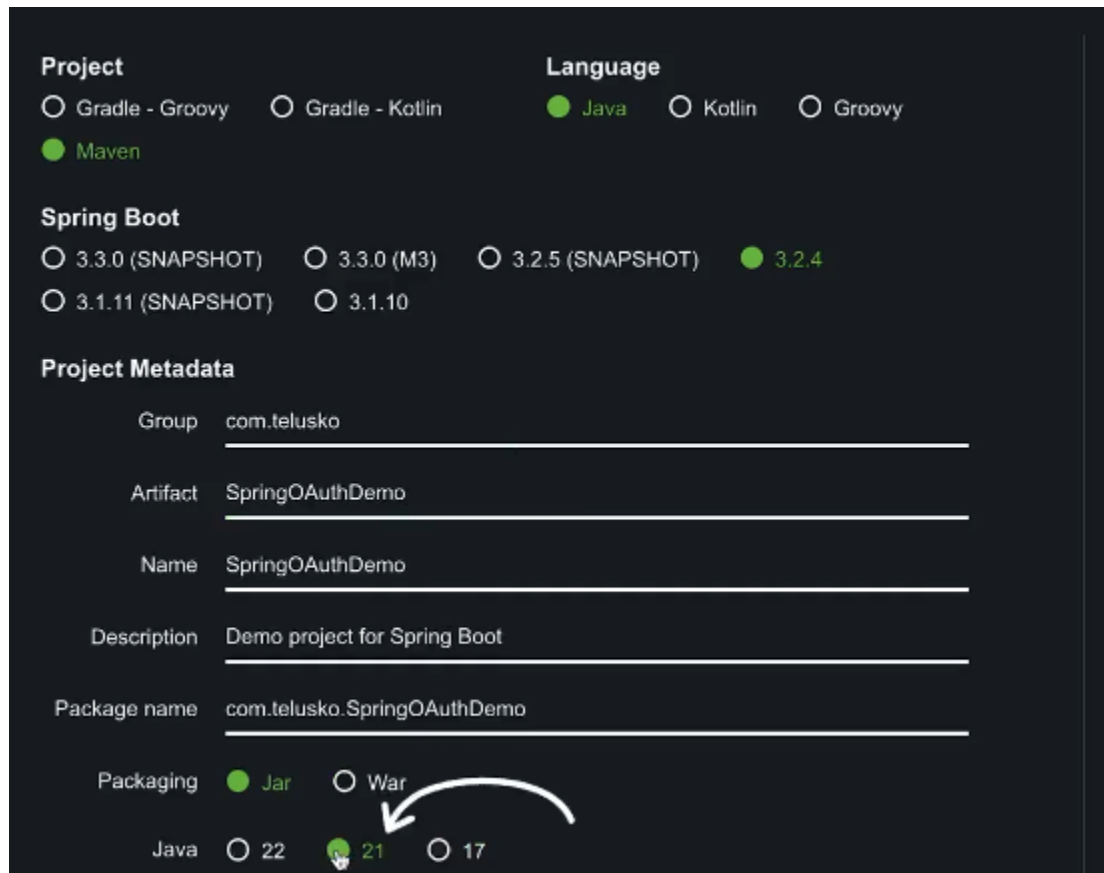
Implementing Google OAuth2 Login in Spring Boot

This guide will help you set up **Google OAuth2 Login** for your Spring Boot application. It includes the configurations and steps required to integrate both providers successfully.

Project Setup

1. Prerequisites

- **Java 17+**
- **Spring Boot 3.0+**
- Maven
- IDE (e.g., IntelliJ, Eclipse)
- Google developer accounts to register your application.



The screenshot shows the Spring Boot project configuration interface. It includes sections for Project, Language, Spring Boot, Project Metadata, and Packaging. The Project section has radio buttons for Gradle - Groovy, Gradle - Kotlin, Java (selected), Kotlin, and Groovy. The Spring Boot section has radio buttons for 3.3.0 (SNAPSHOT), 3.3.0 (M3), 3.2.5 (SNAPSHOT), 3.2.4 (selected), 3.1.11 (SNAPSHOT), and 3.1.10. The Project Metadata section has input fields for Group (com.telusko), Artifact (SpringOAuthDemo), Name (SpringOAuthDemo), Description (Demo project for Spring Boot), and Package name (com.telusko.SpringOAuthDemo). The Packaging section has radio buttons for Jar (selected) and War. At the bottom, there are radio buttons for Java 22, 21 (selected), and 17. A white arrow points from the Java 21 button to the Java 17 button.

Section	Option	Status
Project	Gradle - Groovy	Unselected
	Gradle - Kotlin	Unselected
	Java	Selected
	Kotlin	Unselected
	Groovy	Unselected
Spring Boot	3.3.0 (SNAPSHOT)	Unselected
	3.3.0 (M3)	Unselected
	3.2.5 (SNAPSHOT)	Unselected
	3.2.4	Selected
	3.1.11 (SNAPSHOT)	Unselected
	3.1.10	Unselected
Project Metadata	Group	com.telusko
	Artifact	SpringOAuthDemo
	Name	SpringOAuthDemo
	Description	Demo project for Spring Boot
	Package name	com.telusko.SpringOAuthDemo
Packaging	Jar	Selected
	War	Unselected
Java	22	Unselected
	21	Selected
	17	Unselected

2. Maven Dependencies

Add the following dependencies in your `pom.xml` for Spring Security OAuth2 support:

```
<dependencies>
  <!-- Spring Security OAuth2 -->
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-oauth2-client</artifactId>
  </dependency>
  <!-- Spring Web -->
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
</dependencies>
```

3. Create the Security Configuration

The `SecurityConfig` class configures the security settings and enables OAuth2 login.

Example:

```
package com.telusko.springoauthdemo;

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
                .anyRequest().authenticated() // All requests require authentication
            )
            .oauth2Login(Customizer.withDefaults()); // Enable OAuth2 login
    }
}
```

```
    return http.build();  
  }  
}
```

4. Create a REST Controller

The **HelloController** class defines a simple endpoint for testing OAuth2 authentication.

Example:

```
package com.telusko.springoauthdemo;  
  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
@RestController  
public class HelloController {  
  
    @GetMapping("/hello")  
    public String greet() {  
        return "Welcome to Telusko";  
    }  
}
```

5. Configure **application.properties**

Add your Google OAuth2 credentials in the **application.properties** file. Replace the placeholders with your credentials.

Example:

```
# Application name
spring.application.name=SpringOAuthDemo

# Google OAuth2 credentials
spring.security.oauth2.client.registration.google.client-id=<your-google-client-id>
spring.security.oauth2.client.registration.google.client-secret=<your-google-client-secret>
```

6. Obtain OAuth2 Credentials

Google OAuth2

1. Go to Google Cloud Console.
2. Create a new project or select an existing one.
3. Navigate to **APIs & Services > Credentials**.
4. Create an **OAuth 2.0 Client ID**:
 - Application type: **Web application**
 - Authorized redirect URI:
<http://localhost:8080/login/oauth2/code/google>
5. Copy the **Client ID** and **Client Secret** and add them to your [application.properties](#).

7. Run the Application

1. Start your Spring Boot application by running the [main](#) class.
2. Visit <http://localhost:8080/hello>.
3. You will be redirected to a login page where you can select **Google** for authentication.
4. Once authenticated, you will see the [Welcome to Telusko](#) message.

8. Additional Configuration (Optional)

Custom Redirect After Login

To redirect users to a specific page after login, configure the **DefaultOAuth2UserService**:

```
http
    .oauth2Login(oauth2 -> oauth2
        .defaultSuccessUrl("/hello", true) // Redirect to /hello after login
    );
```

Customizing Login Page

To use a custom login page, add:

```
http
    .oauth2Login(oauth2 -> oauth2
        .loginPage("/custom-login") // Replace with your custom login page endpoint
    );
```

9. Testing

Google Authentication:

- Visit <http://localhost:8080/login/oauth2/code/google>.
- Authenticate using your Google account.

10. Key Components in OAuth2

1. **SecurityFilterChain**: Configures the Spring Security filter chain to enable OAuth2 login.
2. **application.properties**: Stores the OAuth2 client details for Google.
3. **OAuth2 Client**: Spring Security uses **spring-boot-starter-oauth2-client** to handle authentication flows.
4. **Authorized Redirect URIs**: Ensures the authentication server