Connecting with Java

This guide explains how to establish a connection between a Java Spring Boot application and a Keycloak identity provider using the OAuth2 resource server configuration. It walks through the necessary setup, configuration, and creation of a protected endpoint that verifies Keycloak-issued access tokens.

Variables

Certain parameters must be provided to integrate a Spring Boot application with Keycloak. Below is a breakdown of each required variable, its purpose, and where to find it. Here's what each variable represents:

Variable	Description	Purpose
`REALM`	The name of the Keycloak realm	Defines the authentication namespace
`CLIENT_ID`	Client ID from the Keycloak Admin Console	Identifies the Spring Boot app in Keycloak
`ISSUER_URI`	Realm URL (e.g. https://your-domain/realms/your-realm)	Used by Spring Security for token validation
`JWKS_URI`	URL to the JWKS endpoint (auto-resolved by Spring from ISSUER_URI)	Used to fetch public keys for token signature verification

These values can be found in the **Keycloak Admin Console → Clients** and under the **OpenID Connect Endpoints** section for your realm.

Prerequisites

Install Java and Maven

Ensure Java is installed:

```
java -version
```

Ensure Maven is installed:

```
mvn -version
```

If not, download and install from https://adoptium.net or https://maven.apache.org.

Code

Once all prerequisites are set up, create a new Spring Boot project with the following structure:

```
spring-keycloak-demo/
├── src/
│   └── main/
│       ├── java/com/example/demo/
│       │   ├── DemoApplication.java
│       │   └── HelloController.java
│       └── resources/
│           └── application.yml
├── pom.xml
```

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0" ...>
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example</groupId>
  <artifactId>spring-keycloak-demo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <properties>
    <java.version>17</java.version>
    <spring.boot.version>3.1.5</spring.boot.version>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-oauth2-resource-server</artifactId>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
```

```
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
    </plugins>
  </build>
</project>
```

application.yml

```
server:
  port: 8080

spring:
  security:
    oauth2:
      resourceserver:
        jwt:
          issuer-uri: https://your-keycloak-domain/realms/your-realm
```

Replace https://your-keycloak-domain/realms/your-realm with the full issuer URI from your Keycloak realm.

DemoApplication.java

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class DemoApplication {
  public static void main(String[] args) {
    SpringApplication.run(DemoApplication.class, args);
  }
}
```

HelloController.java

```
package com.example.demo;

```

```
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.security.core.annotation.AuthenticationPrincipal;
import org.springframework.security.oauth2.jwt.Jwt;

@RestController
public class HelloController {

  @GetMapping("/")
  public String publicEndpoint() {
    return "Welcome to the public endpoint.";
  }

  @GetMapping("/protected")
  public String protectedEndpoint(@AuthenticationPrincipal Jwt jwt) {
    return "Hello " + jwt.getClaimAsString("preferred_username") + ", you have accessed a
protected route.";
  }
}
```

Execution

1. Start the Spring Boot app with:

```
mvn spring-boot:run
```

2. Generate a JWT access token by logging in through your frontend or REST client (e.g., using Postman with client credentials).
3. Make a request to:

```
GET http://localhost:8080/protected
Authorization: Bearer <access_token>
```

If the token is valid:

- You will receive a welcome message with the Keycloak username
- If no token is provided or it’s invalid, you’ll get a 401 Unauthorized error

---

Revision #1
Created 17 June 2025 09:39:25 by kaiwalya
Updated 17 June 2025 09:50:30 by kaiwalya