## **Authorization Server**

Authorization Server is a supreme architectural component for Web API Security. The Authorization Server acts a centralization authorization point that allows your apps and HTTP endpoints to identify the features of your application.

## **Resource Server**

Resource Server is an application that provides the access token to the clients to access the Resource Server HTTP Endpoints. It is collection of libraries which contains the HTTP Endpoints, static resources, and Dynamic web pages.

## **OAuth2**

OAuth2 is an authorization framework that enables the application Web Security to access the resources from the client. To build an OAuth2 application, we need to focus on the Grant Type (Authorization code), Client ID and Client secret.

## **JWT Token**

JWT Token is a JSON Web Token, used to represent the claims secured between two parties. You can learn more about the JWT token at [www.jwt.io/](https://jwt.io/).

Now, we are going to build an OAuth2 application that enables the use of Authorization Server, Resource Server with the help of a JWT Token.

You can use the following steps to implement the Spring Boot Security with JWT token by accessing the database.

# **Authentication Workflow**

1. The user sends a request to get a token passing his credentials.
2. The server validates the credentials and sends back a token.
3. With every request, the user has to provide the token, and server will validate that token.

We’ll introduce another service called ‘auth service’ for validating user credentials, and issuing tokens.

What about validating the token? Well, it can be implemented in the auth service itself, and the gateway has to call the auth service to validate the token before allowing the requests to go to any service.

Instead, we can validate the tokens at the gateway level, and let the auth service validate user credentials, and issue tokens. And that’s what we’re going to do here.

In both ways, we are blocking the requests unless it’s authenticated (except the requests for generating tokens).

# JSON Based Token (JWT)

A token is an encoded string, generated by our application (after being authenticated) and sent by the user along each request to allow access to the resources exposed by our application.

JSON Based Token (JWT) is a JSON-based open standard for creating access tokens. It consists of three parts; header, payload, and signature.

The header contains the hashing algorithm{type: “JWT”, hash: “HS256”}

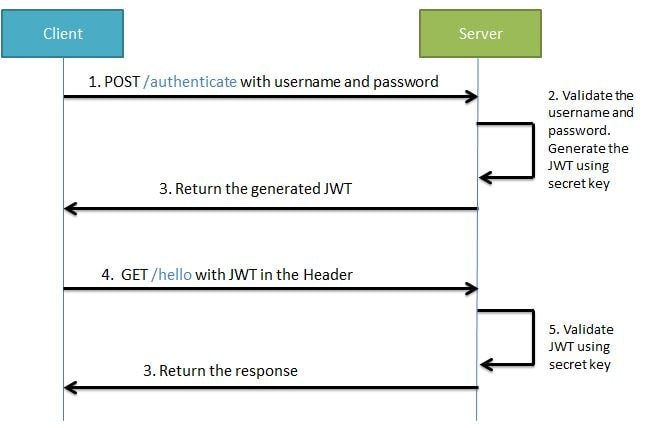
The payload contains attributes (username, email, etc) and their values.

{username: "Omar", email: "omar@example.com", admin: true }

The signature is hashing of: Header + “.” + Payload + Secret key

For better understanding we will be developing the project in stages

* Develop a Spring Boot Application to expose a Simple REST GET API with mapping /hello.
* Configure Spring Security for JWT. Expose REST POST API with mapping /authenticate using which User will get a valid JSON Web Token. And then allow the user access to the api /hello only if it has a valid token



## Spring Security and JWT Configuration

We will be configuring Spring Security and JWT for performing 2 operations-

* **Generating JWT** - Expose a POST API with mapping**/login**. On passing correct username and password it will generate a JSON Web Token(JWT)
* **Validating JWT**  If user tries to access GET API with mapping **/api**. It will allow access only if request has a valid JSON Web Token(JWT)