Spring Framework 6

Beginner to Guru

Overview of OAuth 2 and JWT Tokens

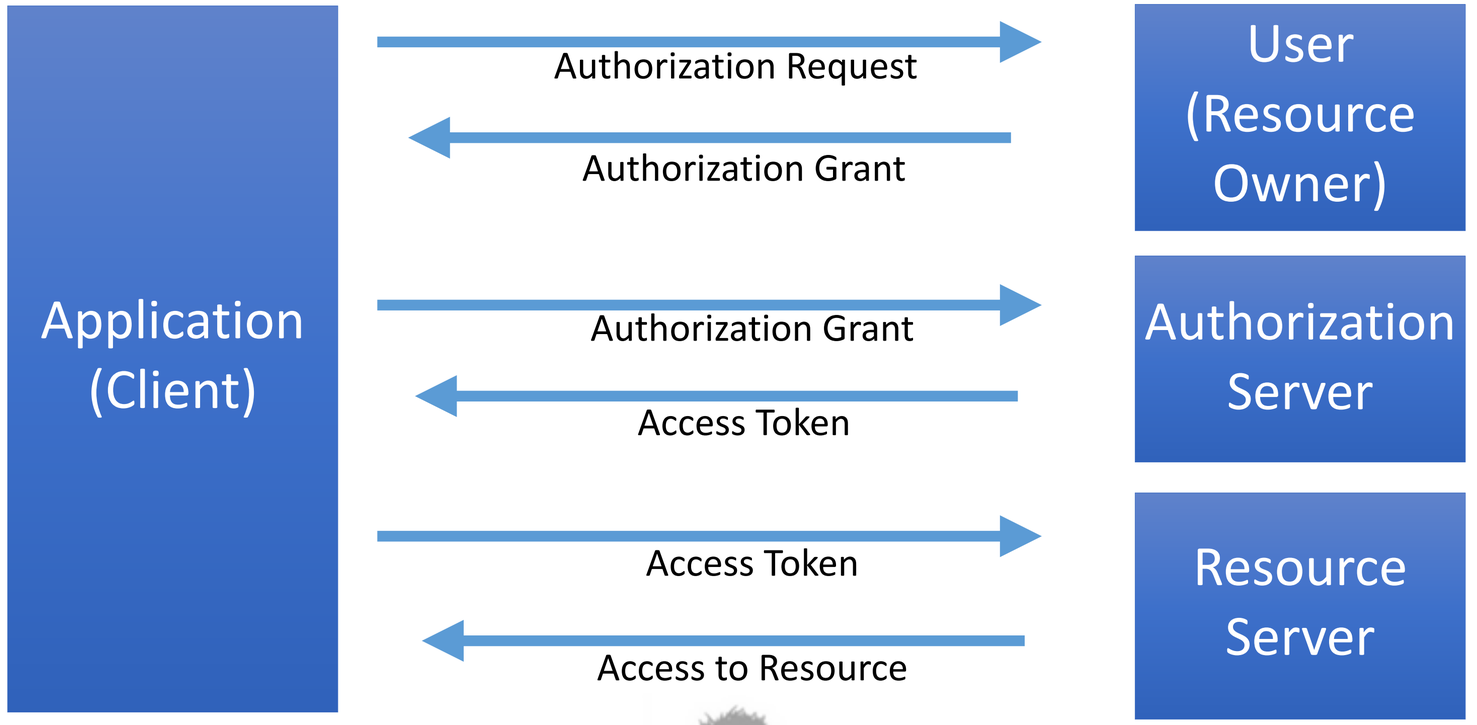
Overview of OAuth 2

- OAuth 2 is an authorization framework
- OAuth 2 is used to grant limited access to resources without full access to the account
- OAuth 2 is used by organizations such as Google, Facebook, and GitHub
  - “Sign in With” - is using OAuth 2
  - Allows you to grant access to a third party application to act on your behalf

OAuth Roles

- **Resource Owner** - the user who wishes to grant an application (client) access
- **Client** - The application requesting access
- **Resource Server** - The resource to access
- **Authorization Server** - Verifies the identity of the user then issues access tokens to the application

Abstract Protocol Flow



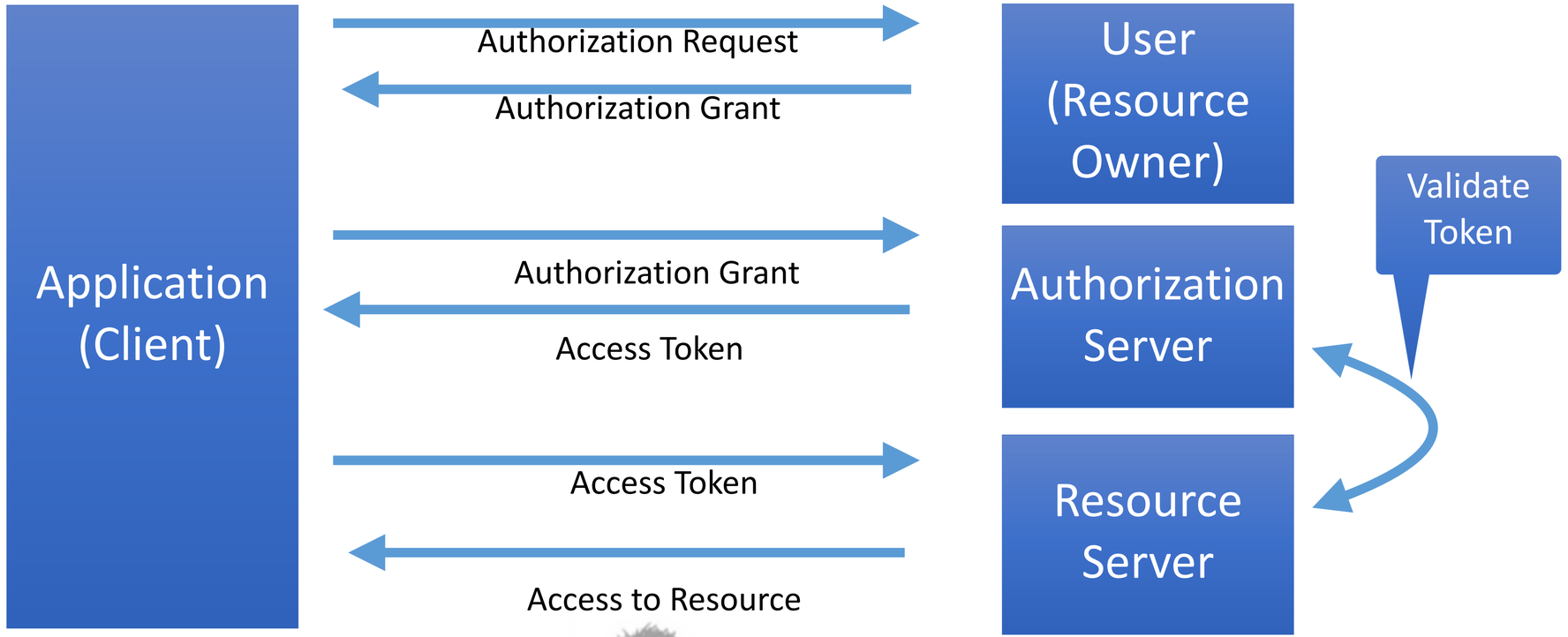
Types of OAuth Authorization Flows

- **Authorization Code Flow** - Server Side web applications where source code is not exposed publicly
- **Client Credentials Flow** - Used by services, where the “user” is a service role
- **Resource Owner Password Flow** - Used by highly trusted applications when redirects cannot be used
- **Implicit Flow** - User grants access with redirects
- **Hybrid Flow** - similar to Client Credentials, but for long running applications

Types of OAuth Authorization Flows (Cont)

- **Device Authorization Flow** - Used by input constrained devices
- **Authorization Code Flow with PKCE** - Flow using proof Key Exchange

Client Credentials Flow w/JWT



JWT Tokens

- **JWT** - JSON Web Token - often pronounced Ja-oot
- RFC 7519 - IETF Specification for JWT, defines how JWT is structured
- HTTP / REST are stateless - each request is self contained
  - Unlike Web Applications which often use session id's stored in cookies
- JWT token has user information and authorized roles (scopes)
- JWT has three parts - Header, Payload (data), and Signature
- The 3 parts are tokenized using Base 64 encoding

JWT Token Signing

- JWT's are signed - which prevents clients from altering the contents of the JWT token
- JWT's maybe signed using a number of techniques
- Symmetric encryption - Uses single key to sign, requires key to be shared
- Asymmetric encryption - Uses public & private key (known as key pair)
  - Private Key is used to generate signature and is not shared
  - Public Key is shared and is used to verify signature

JWT Token Verification

- The Authorization server signs the JWT token using the private key
- The Resource Server requests the public key from the Authorization Server
- Using the public key the resource server verifies the signature of the JWT token
- The resource server can cache the public key for verification of future requests
  - Once the resource server has the public key, JWT tokens can be validated without additional requests from the Authorization Server

OAuth vs HTTP Basic

- HTTP Basic Authentication requires user credentials to be shared with every resource
- HTTP Basic Authentication sends user credentials unencrypted in HTTP Header and can be compromised
- With OAuth user credentials are only shared with Authentication Server
- User credentials cannot be obtained from authorization token
- HTTP Basic Authentication has no concept of security roles
- With OAuth 2 security roles are defined in scopes and passed in token

SPRING FRAMEWORK

GURU