

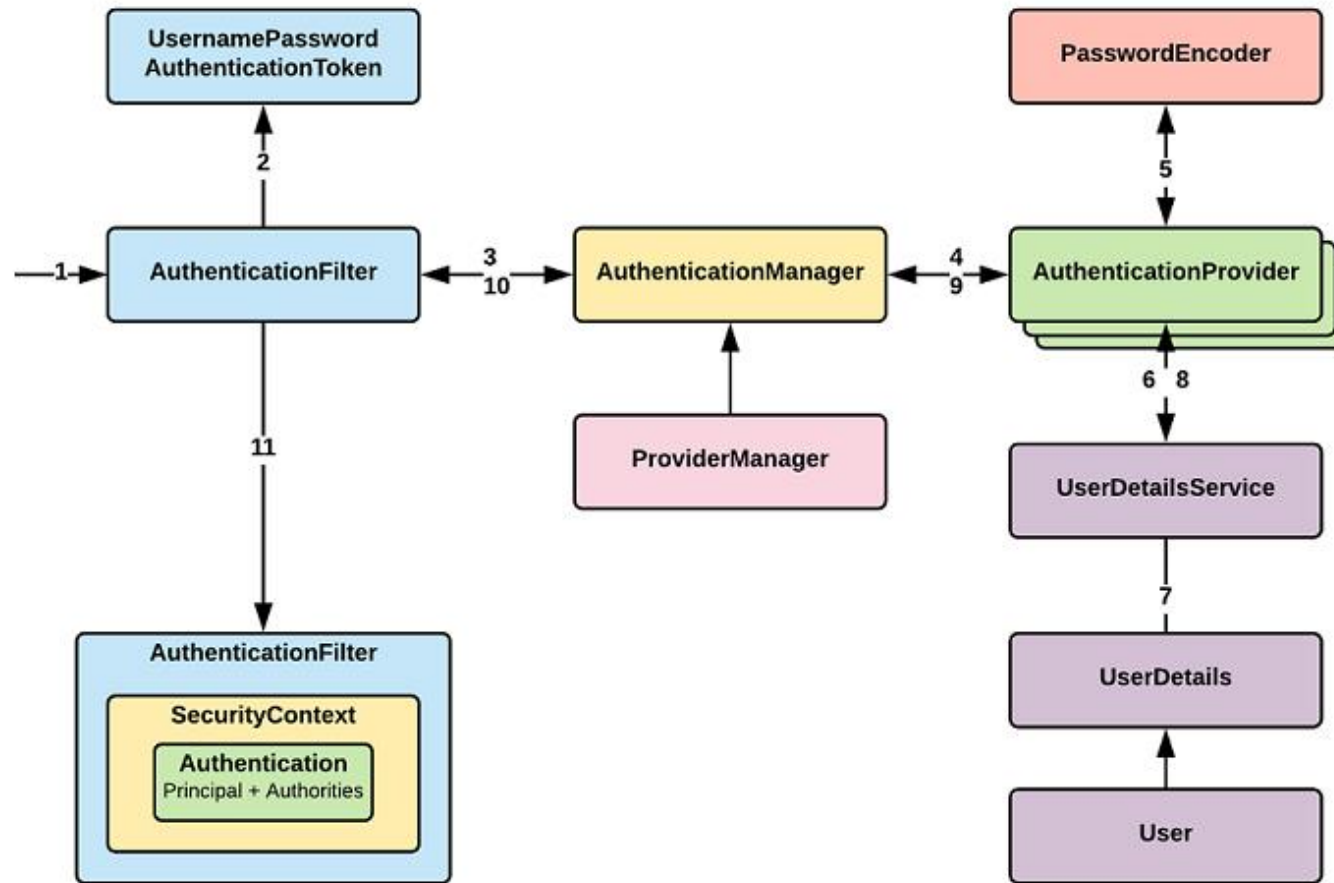
Spring Security & OAuth2

Spring Security & Spring OAuth2

Spring Security + JWT

Spring Security + JWT Configuration

Spring Security + JWT



Spring Security + JWT

Spring Security

Spring Security + JWT

org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter

```
@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

    protected LoginFilter buildLoginFilter(String loginEntryPoint) throws Exception {
        AcceptLoginRequestMatcher matcher = new AcceptLoginRequestMatcher(loginEntryPoint);
        LoginFilter filter = new LoginFilter(successHandler, failureHandler, objectMapper, matcher);
        filter.setAuthenticationManager(this.authenticationManager);
        return filter;
    }

    protected JwtTokenAuthenticationFilter buildJwtTokenAuthenticationFilter(List<String> pathsToSkip, String pattern) throws Exception {
        SkipPathRequestMatcher matcher = new SkipPathRequestMatcher(pathsToSkip, pattern);
        JwtTokenAuthenticationFilter filter = new JwtTokenAuthenticationFilter(failureHandler, tokenExtractor, matcher);
        filter.setAuthenticationManager(this.authenticationManager);
        return filter;
    }

    @Override
    protected void configure(AuthenticationManagerBuilder auth) {
        auth.authenticationProvider ajaxAuthenticationProvider;
        auth.authenticationProvider jwtAuthenticationProvider;
    }

    @Override
    public void configure(WebSecurity web) throws Exception {
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
    }
```

1A

1B

2

3

4

- 1) AuthenticationManager 주입
- 2) AuthenticationProvider 주입
- 3) FilterChainProxy를 생성 (정적 리소스)
- 4) 웹 기반 보안 구성 및 위에서 생성한 필터 추가

Spring Security + JWT

인증 Provider

Spring Security + JWT


org.springframework.security.authentication.AuthenticationProvider

```
@Component
public class LoginAuthenticationProvider implements AuthenticationProvider {

    private CustomPasswordEncoder encoder;
    private UserServiceImpl userService;

    @Autowired
    public LoginAuthenticationProvider(final UserServiceImpl userService, final CustomPasswordEncoder encoder) {
        this.userService = userService;
        this.encoder = encoder;
    }

    ...
}
```



5) 사용자 인증을 위한 UserDetailsService 주입

Spring Security + JWT

JWT Provider

Spring Security + JWT

org.springframework.security.authentication.AuthenticationProvider

```
@Component
public class JwtAuthenticationProvider implements AuthenticationProvider {

    private JwtSettings jwtSettings;

    @Autowired
    public JwtAuthenticationProvider(JwtSettings jwtSettings) {
        this.jwtSettings = jwtSettings;
    }

    ...
}
```

6

6) JWT 설정 클래스 주입

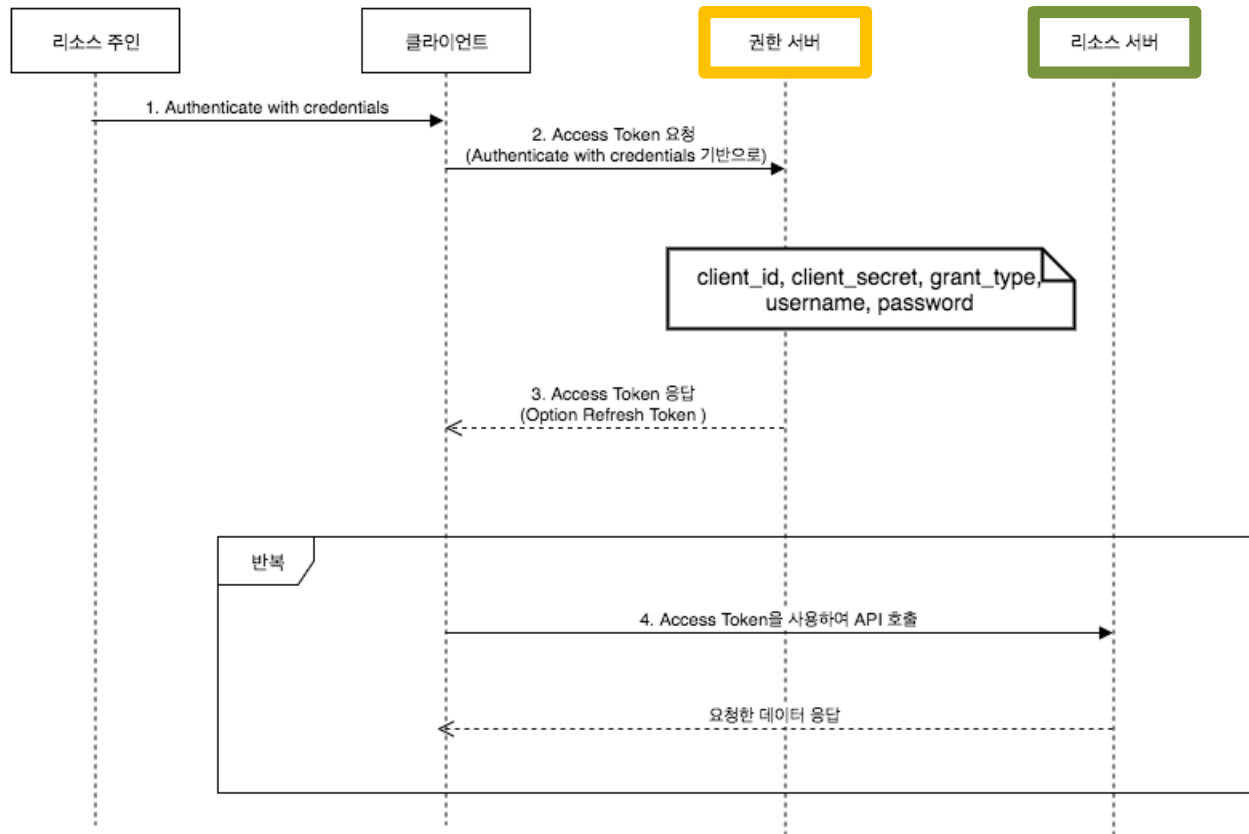
Spring OAuth2 Configuration

Spring OAuth2

공식 문서

<https://tools.ietf.org/html/rfc6749>

Spring OAuth2



Spring OAuth2

Spring Security

Spring OAuth2

org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter

```
@Override
public void configure(WebSecurity web) throws Exception {
    ...
}
```

1

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    ...
}
```

2

```
@Override
@Autowired // <-- This is crucial otherwise Spring Boot creates its own
protected void configure(AuthenticationManagerBuilder auth) throws Exception {
    auth.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder());
}
```

3

- 1) WebSecurity는 FilterChainProxy를 생성하는 필터이다.
web.ignoring().antMatchers("/css/**", "/js/**", "/img/**", "/lib/**");
- 2) HttpSecurity를 통해 HTTP 요청에 대한 웹 기반 보안을 구성할 수 있다.
- 3) AuthenticationManager를 생성하기 위해서는 AuthenticationManagerBuilder를 사용한다.
사용자 인증을 위한 UserDetailsService 서비스를 주입한다.

Spring OAuth2

리소스 서버

Spring OAuth2

org.springframework.security.oauth2.config.annotation.web.configuration.ResourceServerConfigurerAdapter

```
@Configuration
@EnableResourceServer
protected static class ResourceServerConfiguration extends ResourceServerConfigurerAdapter {

    @Override
    public void configure(ResourceServerSecurityConfigurer resources) {
        ...
    }

    @Override
    public void configure(HttpSecurity http) throws Exception {
        ...
    }
}
```

4

5

6

4) API 서버를 OAuth2 인증받게 만들도록 하는 역할을 한다. 기본 옵션은 모든 API의 모든 요청에 대해서 OAuth2 인증을 받도록 한다.

5) OAuth2 리소스 서버 자체의 보안에 대한 정보를 설정한다.

6) HttpSecurity를 통해 HTTP 요청에 대한 웹 기반 보안을 구성할 수 있다.

Spring OAuth2

권한 서버

Spring OAuth2

org.springframework.security.oauth2.config.annotation.web.configuration.AuthorizationServerConfigurerAdapter

```
@Configuration
@EnableAuthorizationServer
protected static class AuthorizationServerConfiguration extends AuthorizationServerConfigurerAdapter {

    @Override
    public void configure(ClientDetailsServiceConfigurer clients) throws Exception {
        ...
    }

    @Override
    public void configure(AuthorizationServerEndpointsConfigurer endpoints) throws Exception {
        ...
    }

    @Override
    public void configure(AuthorizationServerSecurityConfigurer oauthServer) throws Exception {
        ...
    }
}
```

7

8

9

10

7) 액세스 토큰, 리프레시 토큰 발급과 발급된 토큰을 통한 OAuth2 인증 등 핵심기능을 활성화 시킨다. 내부에서는 “/oauth/token”, “/oauth/authorize” 등 OAuth2에서 사용하는 URI의 접근을 활성화한다.

8) ClientDetailsService에 대한 정보를 설정하는 부분.

9) OAuth2 서버가 작동하기 위한 Endpoint에 대한 정보를 설정한다.

10) OAuth2 인증서버 자체의 보안에 대한 정보를 설정한다.

Spring OAuth2

Grant Type

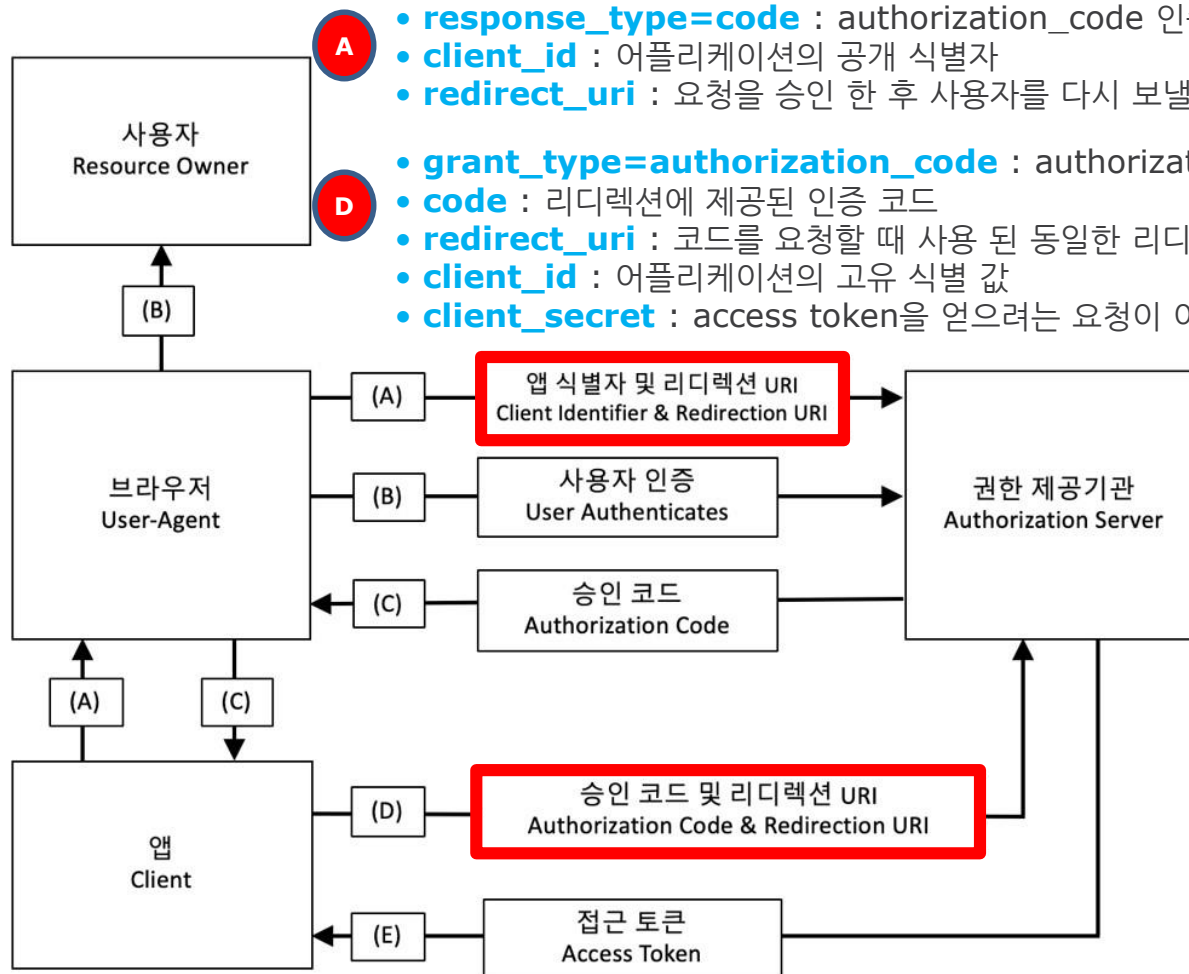
Spring OAuth2

grant_type

사용자가 인증 과정에 개입하는 2가지 방식 (**authorization_code**, **implicit**)과 사용자가 인증 과정에 개입하지 않는 2가지 방식 (**password**, **client_credentials**)이 있다. 마지막으로 만료된 **access_token**을 재발급 받기 위한 **refresh_token**이 있다.

Spring OAuth2

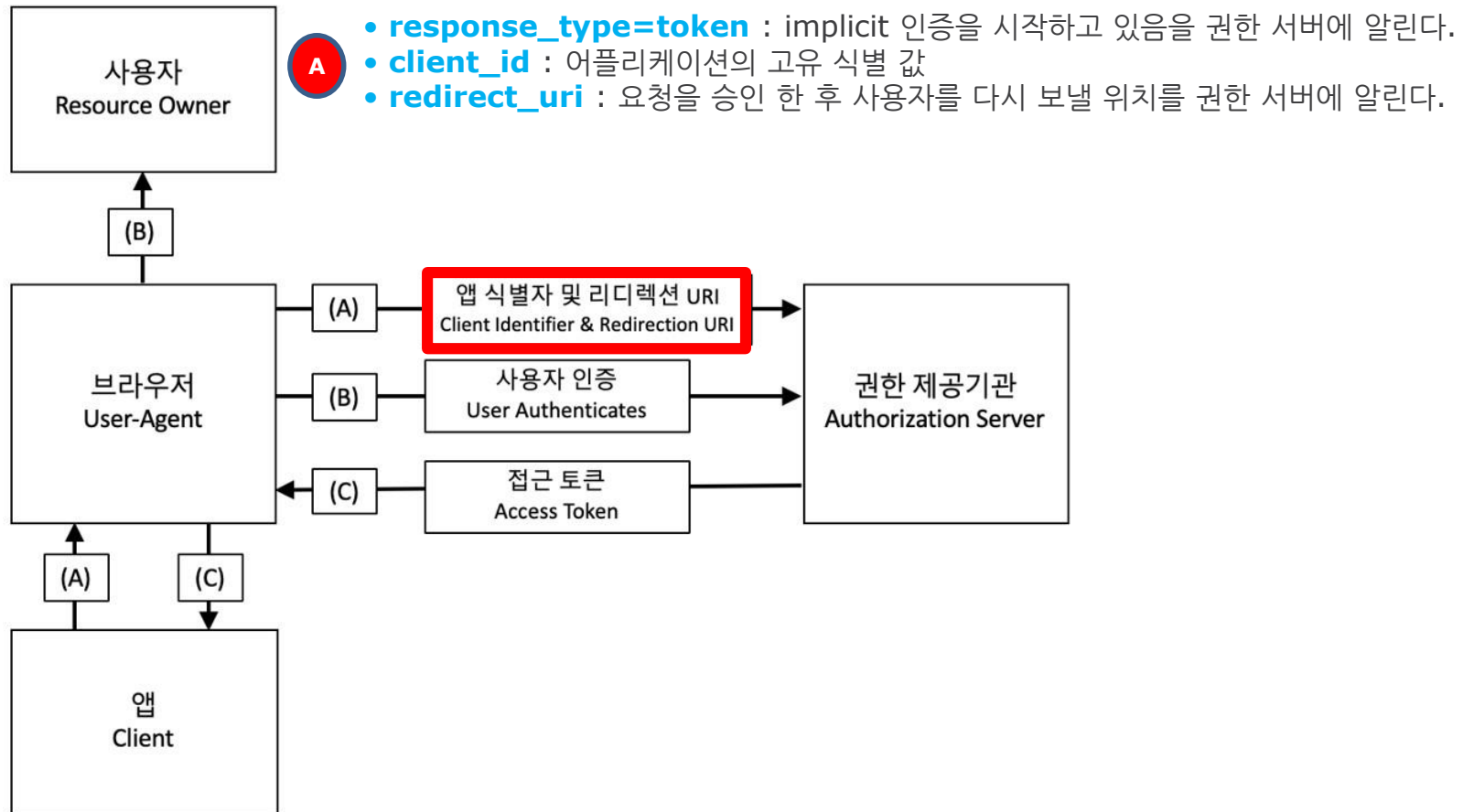
1. authorization_code



- **response_type=code** : authorization_code 인증을 시작하고 있음을 권한 서버에 알린다.
- **client_id** : 어플리케이션의 공개 식별자
- **redirect_uri** : 요청을 승인 한 후 사용자를 다시 보낼 위치를 권한 서버에 알린다.
- **grant_type=authorization_code** : authorization_code 유형을 사용하고 있음을 알린다.
- **code** : 리디렉션에 제공된 인증 코드
- **redirect_uri** : 코드를 요청할 때 사용 된 동일한 리디렉션 URI로 필요하지 않을 경우도 있다.
- **client_id** : 어플리케이션의 고유 식별 값
- **client_secret** : access token을 얻으려는 요청이 어플리케이션에서 이루어진다.

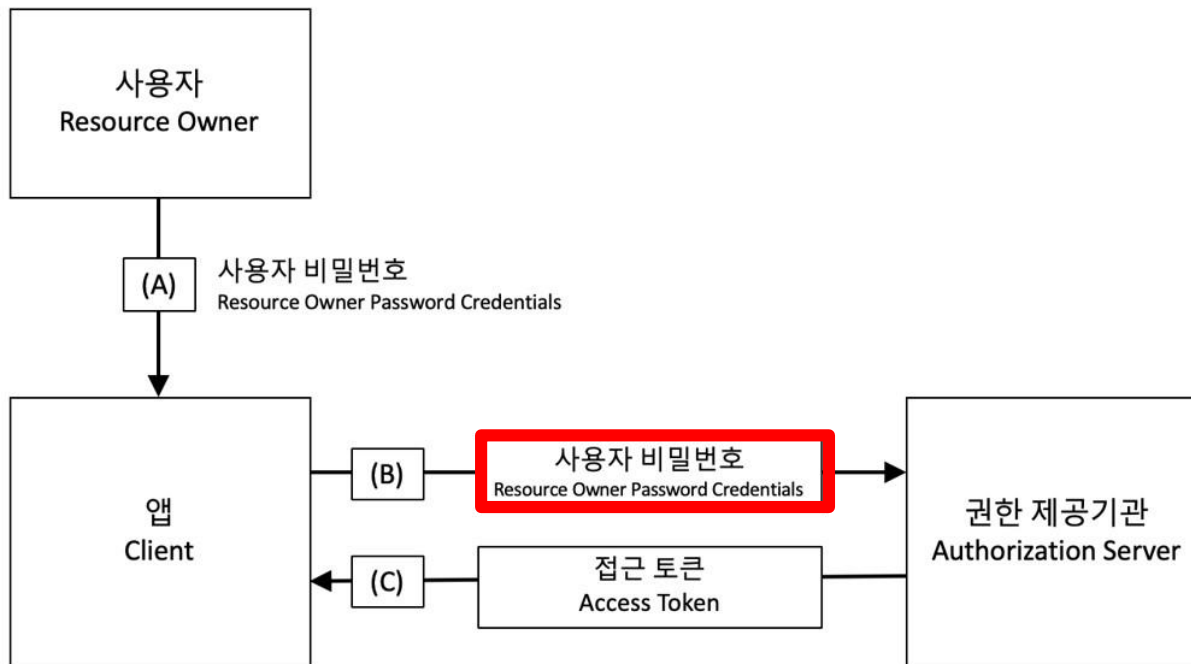
Spring OAuth2

2. implicit



Spring OAuth2

3. password

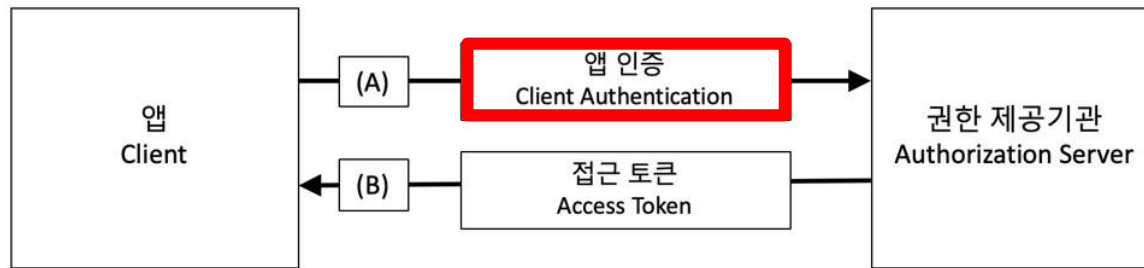


- **grant_type=password** : password 인증을 시작하고 있음을 권한 서버에 알린다.
- **username** : 사용자 이름
- **password** : 사용자의 암호
- **client_id** : 어플리케이션의 고유 식별 값

B

Spring OAuth2

4. client_credentials



A

- **grant_type=client_credentials** : client_credentials 인증을 시작하고 있음을 서버에 알린다.
- **client_id** : 어플리케이션의 고유 식별 값
- **client_secret** : access token을 얻으려는 요청이 어플리케이션에서 이루어진다.