한화생명

한화생명 보험코어시스템 구축

OAuth 2.0 서버구축

Version 0.1

2019.11.05

한화시스템/ICT

제·개정 이력

제·개정일	버전	제·개정 내용	작성자	검토자	승인자
2019.11.05	0.1	제정	김두포		

목 차

1. 개요 4

2. 서버구성도 4

3. 사전지식 5

3.1 OAuth 2.0 구성주체 5

3.2 AccessToken 발급방식 4 가지 6

3.2.1 Authorization Code 방식 6

3.2.2 Implicit 방식 6

3.2.3 Password 방식 (UserDetails Credentails) 6

3.2.4 Client Credentials 방식 6

3.3 OAuth 2.0 Default 제공 API 7

3.3.1 토큰 발급 7

3.3.2 토큰 체크 7

3.3.3 클라이언트 등록 7

3.3.4 클라이언트 승인 7

3.3.5 토큰키 7

4. 서버구축 12

4.1 ClientDetailsServiceConfigurer ..... **오류! 책갈피가 정의되어 있지 않습니다.**

4.2 AuthorizationServerSecurityConfigurer ..... **오류! 책갈피가 정의되어 있지 않습니다.**

4.3 AuthorizationServerEndpointsConfigurer ..... **오류! 책갈피가 정의되어 있지 않습니다.**

5. 구현 및 테스트 8

5.1 Library and Dependencies 8

5.2 DB Schema 11

5.3 테스트 15

5.3.1 토큰발급 15

Authorization Code 방식 16

Implicit 방식 17

Password 방식 (UserDetails Credentails) 18

Client Credentials 방식 19

5.3.2 토큰체크 20

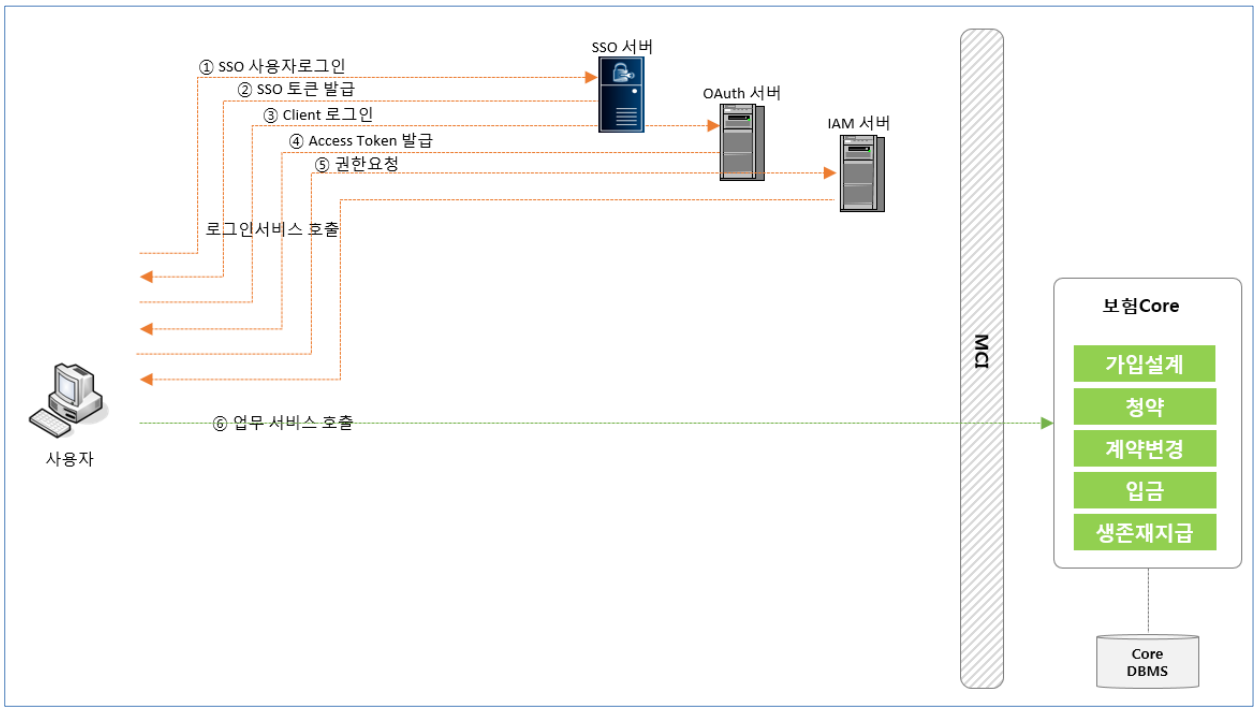
OAuth Server Check 20

JWT Library Check 20

1. 개요

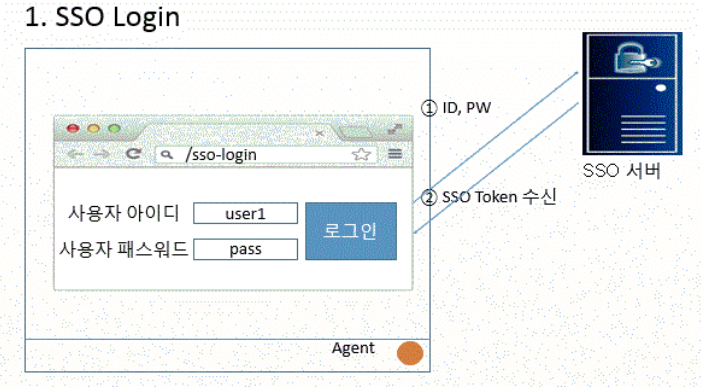
보험코어시스템에서 로그인 및 권한등 업무진입전 전처리에 해당하는 여러 서비스들중 로그인에 이용될 Token 발급서비스에 관해 다룬다.

2. 서버구성도

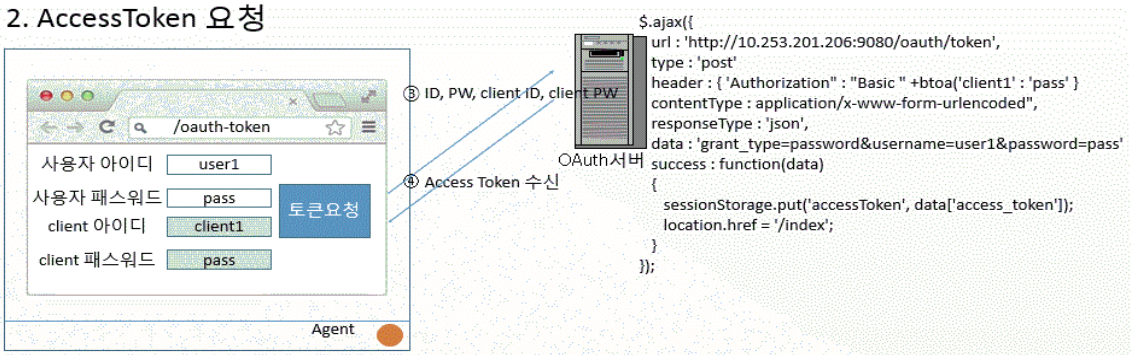


3. 호출흐름

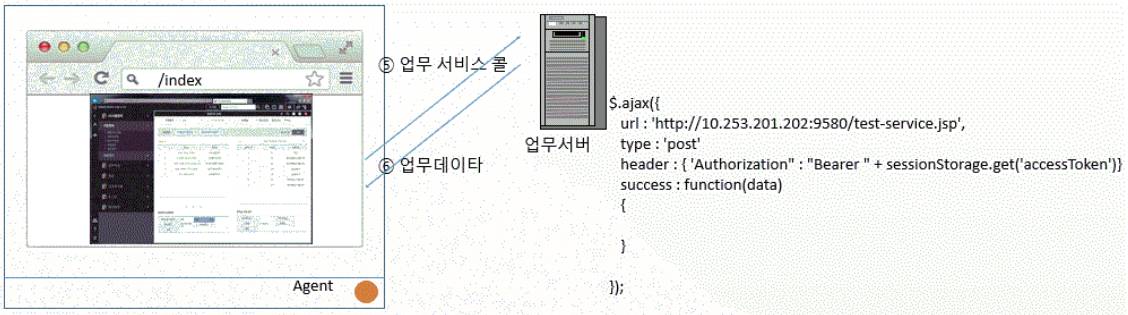
1. SSO Login



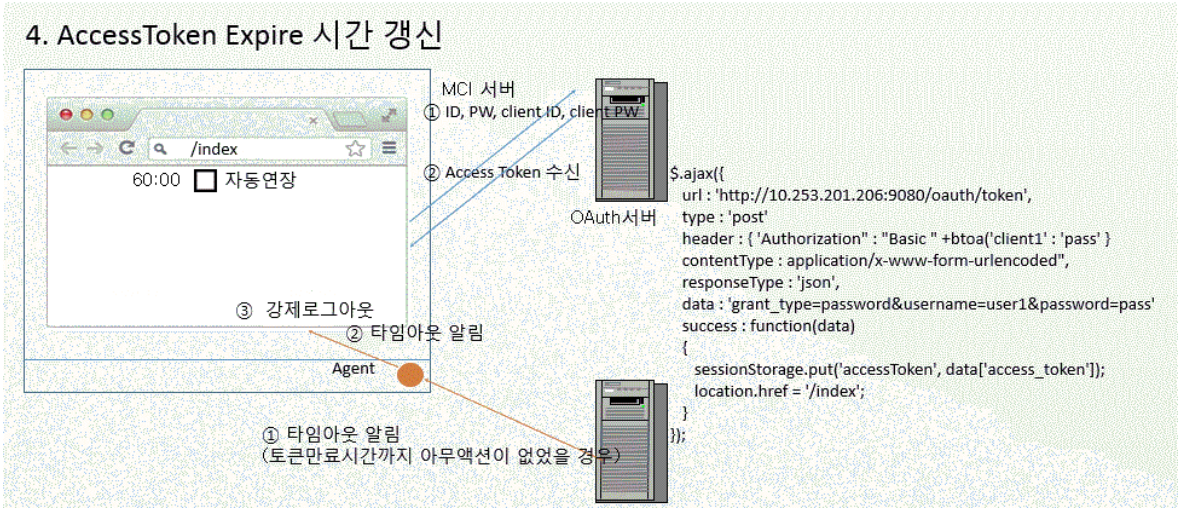
2. AccessToken 요청



3. 업무서비스 Call



4. AccessToken Expire 시간 갱신



4.2 AccessToken 발급방식 4 가지

- 어떤 절차를 통해 AccessToken 을 발급 받을래의 차이.

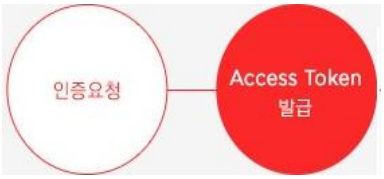
4.2.1 Authorization Code 방식

- 아래 Implicit 방식과 더불어 사용자의 로그인과 권한동의를 요구한다.
(다시말해, Login Page 를 2 개 필요하다. 1 개는 User Login, 1 개는 Client Login)

- User Login 을 먼저 하고,
Client Login 을 한다. (이때 client id, client secret 2 개 다 필요)
토큰을 발급받기 위한 Authorization Code 가 발급되고, 이 코드로 AccessToken 을 발급받는다.

4.2.2 Implicit 방식

- 위 Authorization Code 방식과 더불어 사용자의 로그인과 권한동의를 요구한다.
(다시말해, Login Page 를 2 개 필요하다. 1 개는 User Login, 1 개는 Client Login)



- User Login 을 먼저하고,
Client Login 을 한다. (client id 만 필요. client secret 은 필요없다.)
바로 AccessToken 이 발급된다

4.2.3 Password 방식 (UserDetails Credentails)

- Login Page 로 Redirect 되진 않고, 하드코딩으로 User Login 을 패스할 수 있다.

4.2.4 Client Credentials 방식

- Login Page 로 Redirect 되지 않고, 하드코딩으로도 User Login 정보를 기재할필요가 없다.
Client Login 만 하면 된다.

4.3 OAuth 2.0 Default 제공 API

4.3.1 토큰 발급

/oauth/token

4.3.2 토큰 체크

/oauth/check_token

4.3.3 클라이언트 등록

/oauth/authorize

4.3.4 클라이언트 승인

/oauth/confirm_access

4.3.5 토큰키

/oauth/token_key

※ Customizing
- application.yml

```
oauth:
    paths:
        token: /token
        authorize: /authorize
        confirm: /approve
        check_token: /decode
        token_key: /key
```

5. 구현 및 테스트

5.1 Library and Dependencies

```
<dependencies>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot</artifactId>
			<version>2.1.5.RELEASE</version>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-autoconfigure</artifactId>
			<version>2.1.5.RELEASE</version>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter</artifactId>
			<version>2.1.5.RELEASE</version>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-logging</artifactId>
			<version>2.1.5.RELEASE</version>
		</dependency>

		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-web</artifactId>
			<version>2.1.5.RELEASE</version>
			<exclusions>
				<exclusion>
					<groupId>org.apache.tomcat.embed</groupId>
					<artifactId>tomcat-embed-websocket</artifactId>
				</exclusion>
			</exclusions>
		</dependency>

		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-jdbc</artifactId>
			<version>2.1.5.RELEASE</version>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-devtools</artifactId>
			<version>2.1.5.RELEASE</version>
		</dependency>

		<dependency>
			<groupId>thirdparty.oracle</groupId>
			<artifactId>ojdbc6</artifactId>
			<version>11.2.0.4</version>
		</dependency>

		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-security</artifactId>
			<version>2.1.5.RELEASE</version>
		</dependency>
```

```
57.             <dependency>
58.                 <groupId>org.springframework</groupId>
59.                 <artifactId>spring-context</artifactId>
60.                 <version>5.1.7.RELEASE</version>
61.             </dependency>
62.             <dependency>
63.                 <groupId>org.springframework</groupId>
64.                 <artifactId>spring-core</artifactId>
65.                 <version>5.1.7.RELEASE</version>
66.             </dependency>
67.             <dependency>
68.                 <groupId>org.springframework</groupId>
69.                 <artifactId>spring-aop</artifactId>
70.                 <version>5.1.7.RELEASE</version>
71.             </dependency>
72.             <dependency>
73.                 <groupId>org.springframework</groupId>
74.                 <artifactId>spring-beans</artifactId>
75.                 <version>5.1.7.RELEASE</version>
76.             </dependency>
77.
78.             <dependency>
79.                 <groupId>org.springframework.security</groupId>
80.                 <artifactId>spring-security-jwt</artifactId>
81.                 <version>1.0.10.RELEASE</version>
82.             </dependency>
83.
84.             <dependency>
85.                 <groupId>org.springframework</groupId>
86.                 <artifactId>spring-expression</artifactId>
87.                 <version>5.1.7.RELEASE</version>
88.             </dependency>
89.             <dependency>
90.                 <groupId>org.springframework</groupId>
91.                 <artifactId>spring-jcl</artifactId>
92.                 <version>5.1.7.RELEASE</version>
93.             </dependency>
94.             <dependency>
95.                 <groupId>org.springframework</groupId>
96.                 <artifactId>spring-jdbc</artifactId>
97.                 <version>5.1.7.RELEASE</version>
98.             </dependency>
99.             <dependency>
100.                <groupId>org.springframework.security</groupId>
101.                <artifactId>spring-security-config</artifactId>
102.                <version>5.1.5.RELEASE</version>
103.            </dependency>
104.            <dependency>
105.                <groupId>org.springframework.security</groupId>
106.                <artifactId>spring-security-core</artifactId>
107.                <version>5.1.5.RELEASE</version>
108.            </dependency>
109.            <dependency>
110.                <groupId>org.springframework.security.oauth</groupId>
111.                <artifactId>spring-security-oauth2</artifactId>
112.                <version>2.1.1.RELEASE</version>
113.            </dependency>
114.            <dependency>
115.                <groupId>org.springframework.security</groupId>
116.                <artifactId>spring-security-web</artifactId>
117.                <version>5.1.5.RELEASE</version>
```

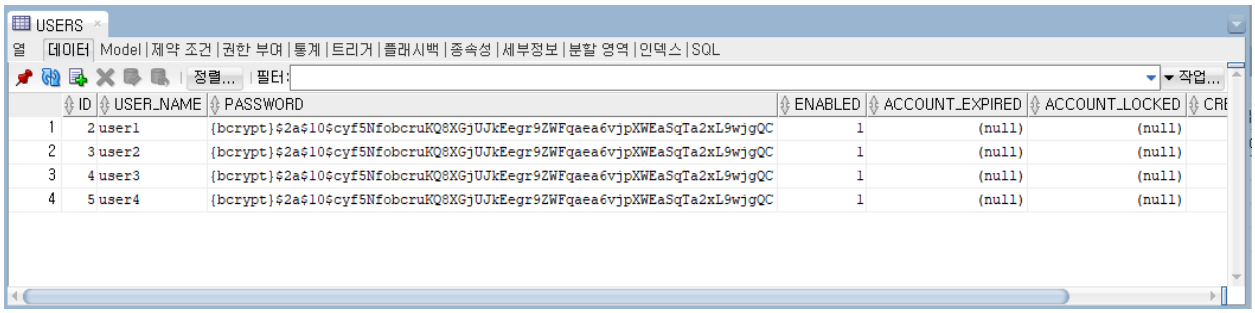
```
118.            </dependency>
119.            <dependency>
120.                    <groupId>org.springframework</groupId>
121.                    <artifactId>spring-tx</artifactId>
122.                    <version>5.1.7.RELEASE</version>
123.            </dependency>
124.            <dependency>
125.                    <groupId>org.springframework</groupId>
126.                    <artifactId>spring-web</artifactId>
127.                    <version>5.1.7.RELEASE</version>
128.            </dependency>
129.            <dependency>
130.                    <groupId>org.springframework</groupId>
131.                    <artifactId>spring-webmvc</artifactId>
132.                    <version>5.1.7.RELEASE</version>
133.            </dependency>
134.            <dependency>
135.                    <groupId>org.mybatis.spring.boot</groupId>
136.                    <artifactId>mybatis-spring-boot-starter</artifactId>
137.                    <version>2.1.0</version>
138.            </dependency>
139.            <dependency>
140.                    <groupId>org.mybatis.spring.boot</groupId>
141.                    <artifactId>mybatis-spring-boot-autoconfigure</artifactId>
142.                    <version>2.1.0</version>
143.            </dependency>
144.            <dependency>
145.              <groupId>jjwt</groupId>
146.              <artifactId>jjwt</artifactId>
147.              <version>0.9.0</version>
148.            </dependency>
149.            <dependency>
150.              <groupId>javax.xml.bind</groupId>
151.              <artifactId>jaxb-api</artifactId>
152.              <version>2.3.1</version>
153.            </dependency>
154.            <dependency>
155.              <groupId>org.apache.tomcat.embed</groupId>
156.              <artifactId>tomcat-embed-jasper</artifactId>
157.              <version>9.0.16</version>
158.              <scope>provided</scope>
159.            </dependency>
160.            <dependency>
161.              <groupId>javax.servlet</groupId>
162.              <artifactId>jstl</artifactId>
163.              <version>1.2</version>
164.            </dependency>
165.    </dependencies>
```

5.2 DB Schema

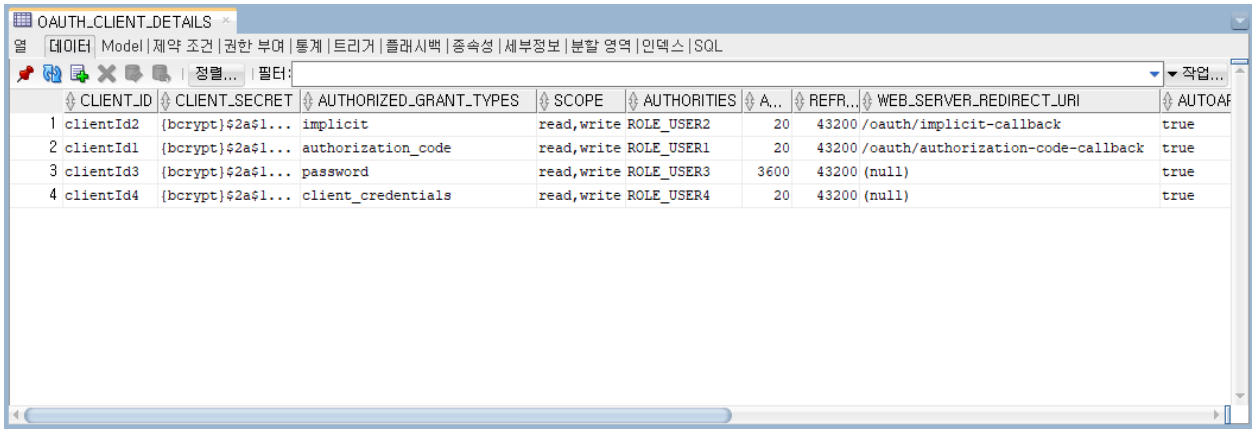
- TokenStore 가 JdbcStore 냐 JwtStore 냐 등에 따라 아래 테이블들이 전체가 쓰이거나 부분적으로 쓰인다.

USERS
USER_AUTHORITY
OAUTH_CLIENT_DETAILS
이외, OAUTH_ACCESS_TOKEN, OAUTH_REFRESH_TOKEN, OAUTH_APPROVALS, OAUTH_CLIENT_TOKEN, OAUTH_CODE

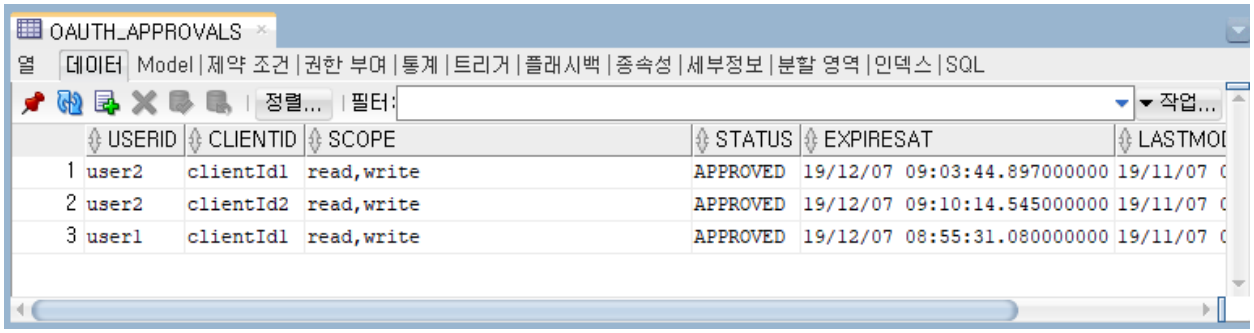
USERS



OAUTH_CLIENT_DETAILS



OAUTH_APPROVALS



5.3 서버구축

5.3.1 CustomUserDetailsService

- 사용자 조회

```
package com.test.oauth2.service;

import java.util.HashSet;

@Service
@Transactional
public class UserDetailsService implements org.springframework.security.core.userdetails.UserDetailsService {

    @Autowired
    private UserDao userDao;

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException
    {
        User user = userDao.findByUsername(username);

        if(user != null)
        {
            UserDetails userDetails = new UserDetails();
            userDetails.setUserName(user.getUserName());
            userDetails.setPassword(user.getPassword());
            List<com.test.oauth2.dto.Authority> list = userDao.findAuthoritiesByUsername(username);

            Set<org.springframework.security.core.GrantedAuthority> authorities = new HashSet<org.springframework.security.core.GrantedAuthority>();
            for (com.test.oauth2.dto.Authority a : list) {
                authorities.add(new GrantedAuthority(a.getName()));
            }
            userDetails.setGrantedAuthorities(authorities);
            return userDetails;
        }
        throw new UsernameNotFoundException(username);
    }
}
```

5.3.2 CustomClientDetailsService

- Client 조회

```
package com.test.oauth2.service;

import java.util.Arrays;

@Service(value="ClientDetailsServiceImpl")
@Transactional
public class ClientDetailsService implements org.springframework.security.oauth2.provider.ClientDetailsService
{
    @Autowired
    private UserDao userDao;

    @Override
    public ClientDetails loadClientByClientId(String clientId) throws ClientRegistrationException
    {
        com.test.oauth2.dto.ClientDetails clientDetails = userDao.findClientById(clientId);

        BaseClientDetails clientDetails1 = new BaseClientDetails();
        clientDetails1.setClientId(clientDetails.getClientId());
        clientDetails1.setClientSecret(clientDetails.getClientSecret());
        clientDetails1.setAuthorizedGrantTypes(Arrays.asList(clientDetails.getAuthorizedGrantTypes()));
        clientDetails1.setScope(Arrays.asList(clientDetails.getScope()));
        clientDetails1.setRegisteredRedirectUri(Collections.singleton(clientDetails.getWebServerRedirectUri()));
        clientDetails1.setResourceIds(Arrays.asList(clientDetails.getResourceIds()));
        Set<SimpleGrantedAuthority> authorities = new HashSet<SimpleGrantedAuthority>();
        authorities.add(new SimpleGrantedAuthority(clientDetails.getAuthorities()));
        clientDetails1.setAuthorities(authorities);

        return clientDetails1;
    }
}
```

5.3.3 AuthServerEndpoint

- 사용자 조회/ Client 조회 서비스 등록, Token 발급/ 체크/ 승인 관련 설정

```
@Override
public void configure(final AuthorizationServerEndpointsConfigurer endpoints)
{
    endpoints
        .authenticationManager(this.authenticationManager)
        .userDetailsService(this.userDetailsService)
        .tokenServices(tokenServices())
        .tokenStore(tokenStore())
        .accessTokenConverter(jwtAccessTokenConverter())
        .userApprovalHandler(userApprovalHandler());
}

@Override
public void configure(final AuthorizationServerSecurityConfigurer oauthServer)
{
    oauthServer
        .passwordEncoder(this.passwordEncoder)
        .tokenKeyAccess("permitAll()")
        .checkTokenAccess("permitAll()");
}

@Bean
public DefaultTokenServices tokenServices()
{
    DefaultTokenServices tokenServices = new DefaultTokenServices();
    tokenServices.setSupportRefreshToken(true);
    tokenServices.setTokenStore(tokenStore());
    tokenServices.setClientDetailsService(clientDetailsService);
    tokenServices.setAuthenticationManager(this.authenticationManager);

    TokenEnhancerChain tokenEnhancerChain = new TokenEnhancerChain();
    tokenEnhancerChain.setTokenEnhancers(Arrays.asList(customAuthTokenEnhancer(), jwtAccessTokenConverter()));
    tokenServices.setTokenEnhancer(tokenEnhancerChain);

    return tokenServices;
}

@Bean
public UserApprovalHandler userApprovalHandler()
{
    ApprovalStoreUserApprovalHandler userApprovalHandler = new ApprovalStoreUserApprovalHandler();
    userApprovalHandler.setClientDetailsService(clientDetailsService);
    userApprovalHandler.setApprovalStore(approvalStore());
    userApprovalHandler.setRequestFactory(requestFactory());

    return userApprovalHandler;
}
```

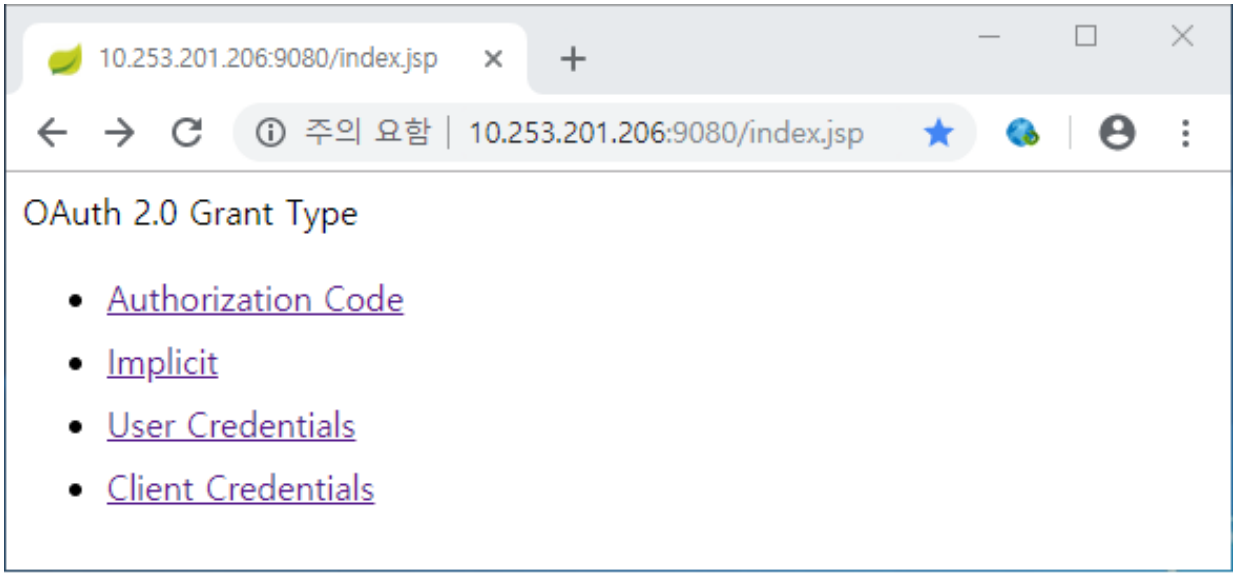
5.3.4 Security

- 로그인 제한 및 CrossDomain 설정

```
@Override
public void configure(HttpSecurity http) throws Exception
{
    http.authorizeRequests().antMatchers("/**/*.js").permitAll().and()
        .authorizeRequests().antMatchers("/**/*.css").permitAll().and()
        .authorizeRequests().antMatchers("/**/*.ico").permitAll().and()
        .authorizeRequests().antMatchers("/**/*.gif").permitAll().and()
        .authorizeRequests().antMatchers("/**/*.jpg").permitAll().and()
        .authorizeRequests().antMatchers("/**/*.png").permitAll().and()
        .authorizeRequests().antMatchers("/**/*.htm").permitAll().and()
        .authorizeRequests().antMatchers("/**/*.html").permitAll().and()
        .authorizeRequests().antMatchers("/index.jsp").permitAll().and()
        .authorizeRequests().antMatchers("/test*.jsp").permitAll().and()
        .formLogin().permitAll().and()
        .logout().permitAll().and()
        .authorizeRequests().antMatchers("/oauth/**").permitAll().and().cors().and().csrf().disable()
        .authorizeRequests().antMatchers("/**/*").access("hasRole('ROLE_USER')").and().cors().and().csrf().disable()
        .sessionManagement().maximumSessions(1);
}

@Bean
public CorsConfigurationSource corsConfigurationSource()
{
    CorsConfiguration config = new CorsConfiguration();
    config.setAllowedOrigins(Arrays.asList(new String[]{"http://localhost",
                                                        "http://localhost:9080",
                                                        "http://localhost:8080",
                                                        "http://localhost:52194",
                                                        "http://10.253.201.206:1080",
                                                        "http://10.253.201.202:9580"}));
    config.addAllowedMethod("*");
    config.addAllowedHeader("*");
    config.setAllowCredentials(true);
    config.setMaxAge(3600L);

    UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();
    source.registerCorsConfiguration("/**", config);

    return source;
}

@SuppressWarnings("rawtypes")
@Bean
public FilterRegistrationBean corsFilteringRegistrationBean()
{
    @SuppressWarnings("unchecked")
    FilterRegistrationBean bean = new FilterRegistrationBean(new CorsFilter(corsConfigurationSource()));
    bean.setOrder(0);
    return bean;
}
```

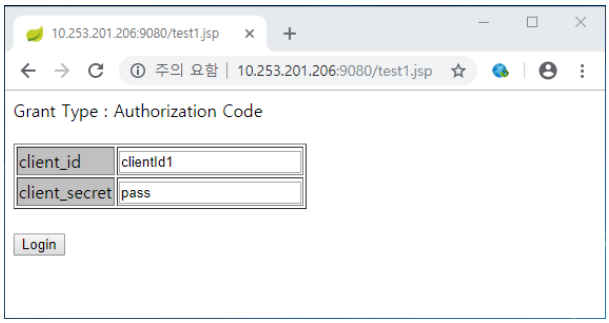
5.4 테스트

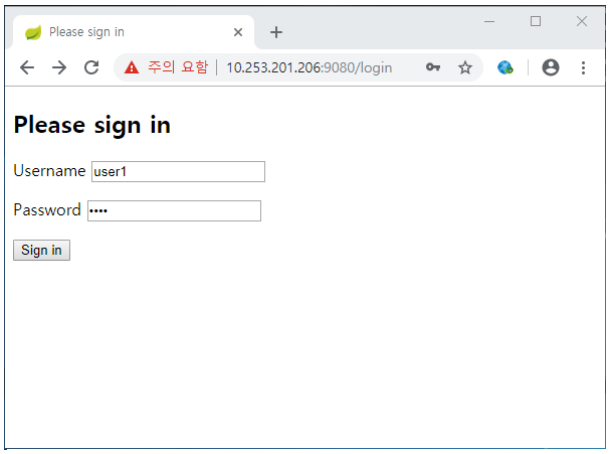
5.4.1 토큰발급

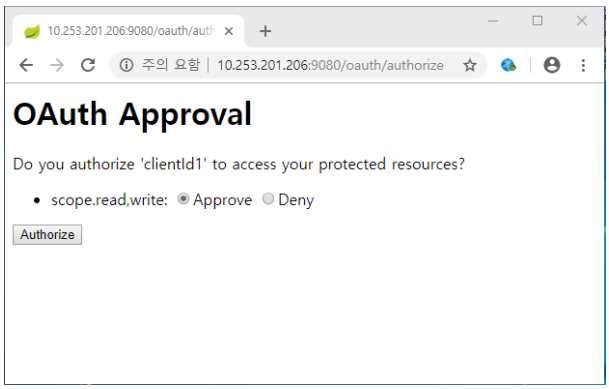


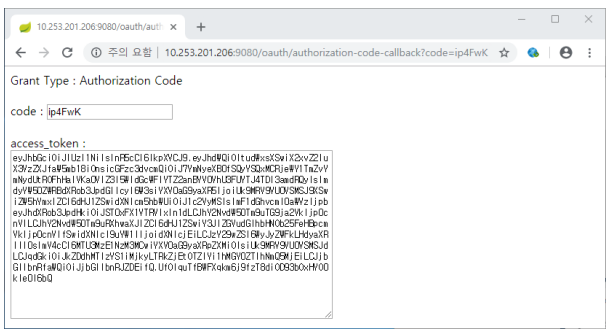
1. Authorization Code 방식

http://10.253.201.206:9080/test1.jsp



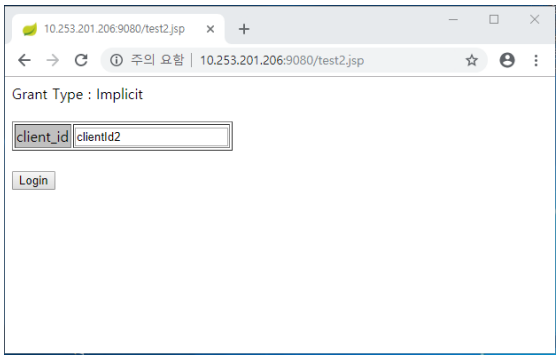


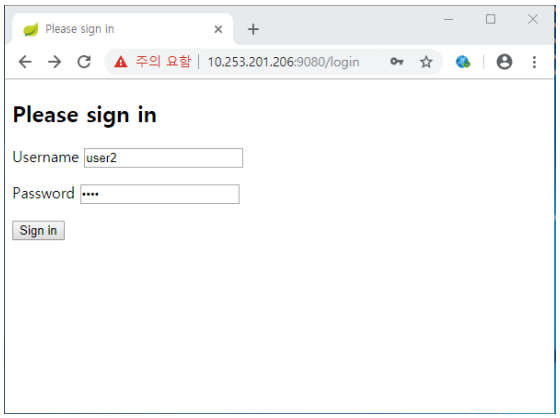


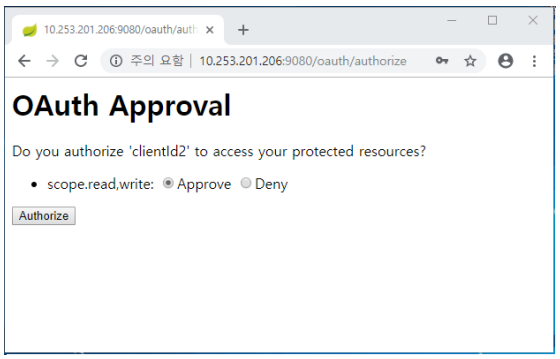


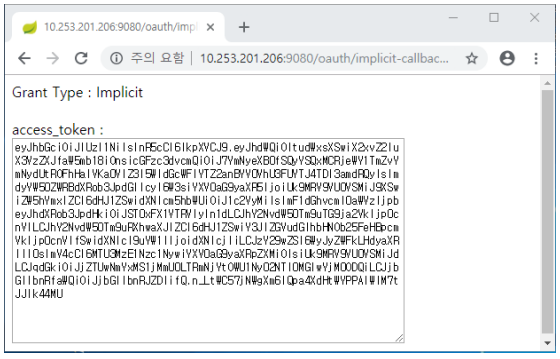
2. Implicit 방식

http://10.253.201.206:9080/test2.jsp



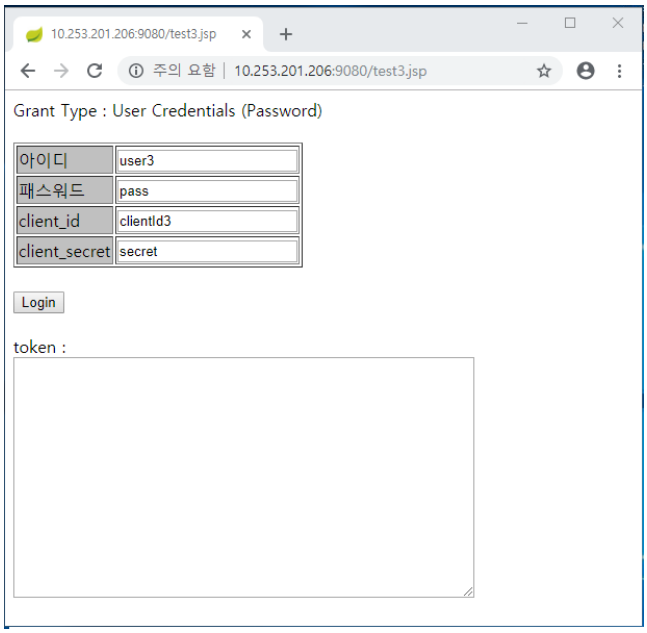


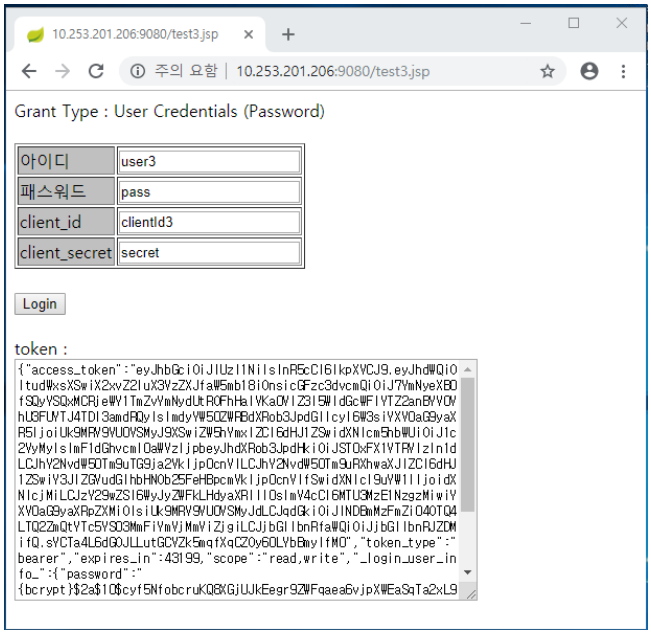




3. Password 방식 (UserDetails Credentails)

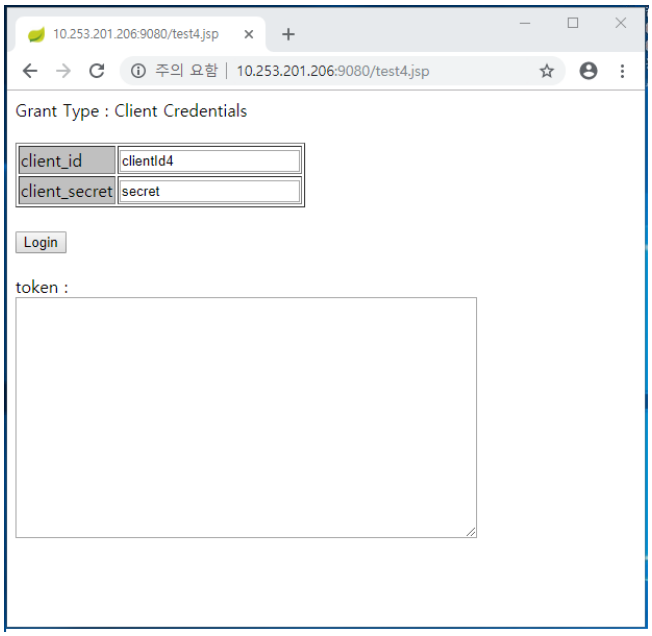
http://10.253.201.206:9080/test3.jsp

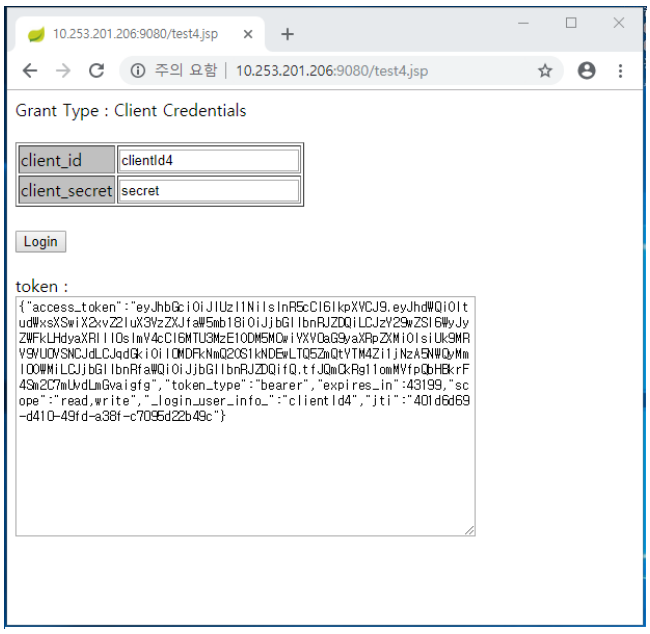




4. Client Credentials 방식

http://10.253.201.206:9080/test4.jsp

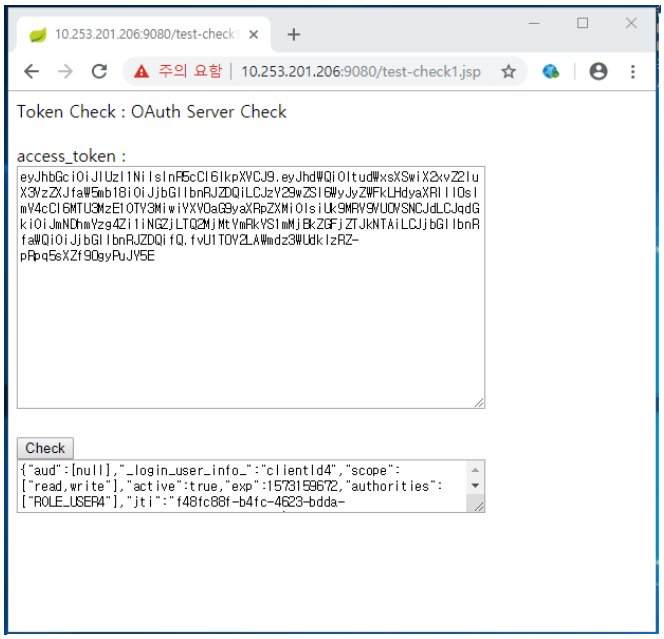




5.4.2 토큰체크

1. OAuth Server Check

http://10.253.201.206:9080/test-check1.jsp



2. JWT Library Check

http://10.253.201.206:9080/test-check2.jsp