

Quiz Questions

(1) OAuth vs OIDC

← Classroom

OAuth and OIDC
OAuth vs OIDC

?

V

Problem Submissions Doubts

✓ OAuth vs OIDC

Easy • Score 20/20

Problem statement

Which of the following is a notable difference between OAuth and OIDC?

Send feedback

Options: Pick one correct answer from below

☒ OAuth is used for access delegation, while OIDC is used for user authentication. ✓

☐ OAuth and OIDC are interchangeable and serve the same purpose.

☐ OAuth is exclusively used for user authentication, while OIDC is only for authorization.

☐ OAuth and OIDC are identical and have no differences in their functionalities.

Solution description

OAuth grants authorization and access tokens to third-party applications, enabling them to access resources without revealing user credentials. OIDC, however, focuses on user authentication by providing identity information via tokens.

(2) Client in OAuth

← Classroom

OAuth and OIDC
Client in OAuth

?

V

Problem Submissions Doubts

✓ Client in OAuth

Easy • Score 20/20

Problem statement

Who is the "client" in OAuth 2.0?

Send feedback

Options: Pick one correct answer from below

☐ The end-user requesting access to resources

☐ The server hosting the protected resources

☒ The application requesting access to the protected resources on behalf of the end-user ✓

☐ The authorization server validating user credentials

Solution description

The client in OAuth 2.0 refers to the application that requests access to resources (protected data) on behalf of the end-user. It could be a web app, mobile app, or service.

(3) Okta

← Classroom

OAuth and OIDC
Okta

?

V

Problem Submissions Doubts

✓ Okta

Easy • Score 20/20

Problem statement

What does Okta primarily offer as a service?

Send feedback

Options: Pick one correct answer from below

☐ Content Delivery Network (CDN)

☒ Identity as a Service (IDaaS) ✓

☐ Cloud Storage

☐ Customer Relationship Management (CRM)

Solution description

Okta primarily provides identity and access management solutions, including user authentication, single sign-on (SSO), and access control.

(4) Groups in Okta

← Classroom

OAuth and OIDC
Groups in Okta

?

V

Problem Submissions Doubts

✓ Groups in Okta

Easy • Score 20/20

Send feedback

Problem statement

What is the primary purpose of **groups** Okta's identity and access management system?

Options: Pick one correct answer from below

☐ To organize users as resource owners and clients.

☐ To classify users based on their names.

☒ To categorise users according to their roles or any specific category.

☐ To segregate users by their device types.

Solution description

Groups in Okta categorize users based on specific criteria or roles, simplifying access management by collectively applying policies and permissions to groups.

(5) URL for Okta Login

← Classroom

OAuth and OIDC
URL for Okta Login

?

V

Problem Submissions Doubts

✓ URL for Okta Login

Easy • Score 20/20

Send feedback

Problem statement

What is the URL typically used for Okta login?

Options: Pick one correct answer from below

☐ /okta/login

☐ /oauth/authorization/login

☐ /login/okta

☒ /oauth2/authorization/okta

Solution description

The correct URL for initiating the login process via Okta in an OAuth 2.0 setup often involves the /oauth2/authorization/okta endpoint. This URL starts the OAuth 2.0 authorization flow with Okta as the authorization server, allowing users to log in and provide consent to access protected resources.

(6) Todo Application

✓ Todo Application

Easy • Score 20/20

Send feedback

Problem statement

Suppose you are making a secured todo Application using OAuth with Okta authorization. In the application, you have a public form login API: **"/customer/login"**. Choose the correct implementation of the filterChain bean for configuring OAuth for this API.

Options: Pick one correct answer from below

☐

```
@Bean
public SecurityFilterChain #filterChain(HttpSecurity http) throws Exception
{
    http
        .csrf().disable()
        .authorizeHttpRequests()
        .formLogin()
        .loginPage("/customer/login").permitAll()
        .and()
        .oauth2Login()
        .loginPage("/customer/login");
    return http.build();
}
```

☐

```
@Bean
public SecurityFilterChain #filterChain(HttpSecurity http) throws Exception
{
    http
        .csrf().disable()
        .authorizeHttpRequests()
        .formLogin()
        .loginPage("/customer/login").permitAll()
        .and()
        .oauth2Login()
        .loginPage("/customer/login");
    return http.build();
}
```

☒

```
@Bean
public SecurityFilterChain #filterChain(HttpSecurity http) throws Exception
{
    http
        .csrf().disable()
        .authorizeHttpRequests()
        .and()
        .formLogin()
        .loginPage("/customer/login").permitAll()
        .and()
        .oauth2Login()
        .loginPage("/customer/login");
    return http.build();
}
```

(7) Todo Application II

← Classroom

OAuth and OIDC
Todo Application II

Problem Submissions Doubts

✓ Todo Application II

Easy • Score 20/20

Problem statement

Continuing the todo application, we must implement a new API, `/auth/login`, to retrieve the access token when signing in with Okta.
The AuthResponse DTO is given below, in which we will wrap all the access token details.

```
@Data
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
public class AuthResponse {

    private String userId;
    private String accessToken;
    private long expireAt;
    private Collection<String> authorities;
}
```

Choose the correct implementation of the `/auth/login` API.

Options: Pick one correct answer from below

☒

```
...
@GetMapping("/login")
public ResponseEntity<AuthResponse> login(
    @AuthenticationPrincipal OAuth2AuthorizedClient oAuth2AuthorizedClient,
    @RequestBody User user) {

    AuthResponse authResponse = new AuthResponse();
    authResponse.setUserId(user.getId());
    authResponse.setAccessToken(oAuth2AuthorizedClient.getAccessToken().getAccessToken());
    authResponse.setExpiresAt(oAuth2AuthorizedClient.getExpiresAt());
    authResponse.setAuthorities(user.getAuthorities().stream().map(GrantedAuthority::getAuthority).collect(Collectors.toList()));
    return new ResponseEntity<AuthResponse>(authResponse, HttpStatus.OK);
}
```

☐

```
...
@GetMapping("/login")
public ResponseEntity<AuthResponse> login(
    @AuthenticationPrincipal OAuth2AuthorizedClient oAuth2AuthorizedClient,
    @RequestBody User user) {

    AuthResponse authResponse = new AuthResponse();
    authResponse.setUserId(user.getId());
    authResponse.setAccessToken(oAuth2AuthorizedClient.getAccessToken().getAccessToken());
    authResponse.setExpiresAt(oAuth2AuthorizedClient.getExpiresAt());
    authResponse.setAuthorities(user.getAuthorities().stream().map(GrantedAuthority::getAuthority).collect(Collectors.toList()));
    return new ResponseEntity<AuthResponse>(authResponse, HttpStatus.OK);
}
```

☐

```
...
@GetMapping("/login")
public ResponseEntity<AuthResponse> login(
    @AuthenticationPrincipal OAuth2AuthorizedClient oAuth2AuthorizedClient,
    @RequestBody User user) {

    AuthResponse authResponse = new AuthResponse();
    authResponse.setUserId(user.getId());
    authResponse.setAccessToken(oAuth2AuthorizedClient.getAccessToken().getAccessToken());
    authResponse.setExpiresAt(oAuth2AuthorizedClient.getExpiresAt());
    authResponse.setAuthorities(user.getAuthorities().stream().map(GrantedAuthority::getAuthority).collect(Collectors.toList()));
    return authResponse;
}
```

(8) Okta Integration

← Classroom

OAuth and OIDC
Okta Integration

Problem Submissions Doubts

✓ Okta Integration

Easy • Score 20/20

Problem statement

Which of the following statements about integrating Okta with a Spring Boot application is correct?

Options: Pick one correct answer from below

☐ Okta authentication will allow seamless access to user credentials stored in the local database without additional configuration.

☒ Okta and the local database are separate authentication sources and require separate authentication flows.

☐ Integrating Okta means migrating all user credentials to Okta's database for authentication.

☐ Okta integration automatically syncs and updates user credentials stored in the local database.

Solution description

Integrating Okta for authentication in a Spring Boot application involves using Okta as an identity provider while maintaining the local database for additional user management. Users authenticated via Okta will use Okta's authentication flow, while users in the local database will authenticate separately, utilising the application's configured authentication mechanism.

(9) Sign In with Google

← Classroom

OAuth and OIDC
Sign in with Google

?

V

Problem Submissions Doubts

✓ Sign in with Google

Easy • Score 20/20

Problem statement

Where are the user credentials saved when implementing "Sign in with Google"?

Send feedback

Options: Pick one correct answer from below

☐ Google servers

☒ Local database managed by the application

☐ Client Database

☐ OAuth server

Solution description

When integrating "Sign in with Google" and storing user credentials in a local database, the responsibility for managing and storing these credentials lies with the application itself.

(10) Todo Application III

← Classroom

OAuth and OIDC
Todo Application III

?

V

Problem Submissions Doubts

✓ Todo Application III

Easy • Score 20/20

Problem statement

Continuing the todo Application, we are implementing "sign-in with Google". Choose the correct implementation of the filterChain bean for the same.

Send feedback

Options: Pick one correct answer from below

☒

```
***
@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception
{
    http
        .csrf().disable()
        .authorizeHttpRequests()
        .formLogin()
        .loginPage("/customer/login").permitAll()
        .and()
        .oauth2Login()
        .loginPage("/customer/login")
        .userInfoEndpoint()
        .oidcUserService(this.oidcUserService());
    return http.build();
}
```

☐

```
***
@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception
{
    http
        .csrf().disable()
        .authorizeHttpRequests()
        .formLogin()
        .loginPage("/customer/login").permitAll()
        .and()
        .oauth2Login()
        .loginPage("/customer/login")
        .return http.build();
}
```

☐

```
***
@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception
{
    http
        .csrf().disable()
        .authorizeHttpRequests()
        .formLogin()
        .loginPage("/customer/login").permitAll()
        .and()
        .oauth2Login()
        .userInfoEndpoint()
        .oidcUserService(this.oidcUserService());
        return http.build();
}
```

☐ None of the above

(11) Todo Application IV

Problem

Submissions

Doubts

Todo Application IV

Easy • Score 20/20

Send feedback

Problem statement

Continuing the previous question, now we must implement the `oidcUserService()` method to tell the `filterChain` bean to manage the OIDC user from the Database. Choose the correct implementation of the `oidcUserService()` method.

Options: Pick one correct answer from below

☒

```
public OAuth2UserService<OAuth2Request, OAuth2User> oidcUserService() {  
    return new OAuth2User() {  
        @Override public OAuth2Request getOAuth2Request() {  
            return null;  
        }  
        @Override public OAuth2User getOAuth2User() {  
            return null;  
        }  
    };  
}
```

☐

```
public OAuth2UserService<OAuth2Request, OAuth2User> oidcUserService() {  
    return new OAuth2User() {  
        @Override public OAuth2Request getOAuth2Request() {  
            return null;  
        }  
        @Override public OAuth2User getOAuth2User() {  
            return null;  
        }  
    };  
}
```

☐

```
public OAuth2UserService<OAuth2Request, OAuth2User> oidcUserService() {  
    return new OAuth2User() {  
        @Override public OAuth2Request getOAuth2Request() {  
            return null;  
        }  
        @Override public OAuth2User getOAuth2User() {  
            return null;  
        }  
    };  
}
```

☐

None of the above

Solution description

This method `oidcUserService()` is an override implementing `OAuth2UserService` for OIDC (OpenID Connect) authentication. It intercepts the OIDC user request, delegates to the default `OAuth2UserService` for user details, fetches the user from the local database using the email attribute obtained from Google authentication, and constructs a custom `DefaultOAuth2User` object with the retrieved user's authorities and ID token for further authentication and authorization in the Spring Security framework.