

Quiz Questions

(1) Use case for user persistence

← Classroom

User Persistence & JWT Authentication
Use Cases for User Persistence

?

V

Problem Submissions Doubts

✔ usecase for user persistence

Easy • Score 20/20

Send feedback

Problem statement

Which of the following is **NOT** a typical use case for user data persistence?

Options: Pick one correct answer from below

Attempts left: 0/2

☐ Saving user preferences and settings.

☐ Remembering a user's login status.

☒ Storing sensitive information

☐ Keeping track of a user's shopping cart items

Solution description

Typically, user data persistence is not used to store sensitive information, as it can pose security risks. Sensitive data, such as passwords and financial information, should be securely stored using appropriate encryption and security measures rather than simple data persistence. Storing sensitive information insecurely can lead to data breaches and compromise user privacy and security.

(2) Purpose of Storing Credentials

← Classroom

User Persistence & JWT Authentication
Purpose of Storing Credentials

?

V

Problem Submissions Doubts

✔ Purpose of Storing credentials

Easy • Score 20/20

Send feedback

Problem statement

What is the primary purpose of securely storing user credentials in a database?

Options: Pick one correct answer from below

Attempts left: 1/2

☐ To provide a convenient way for users to share their passwords

☐ To make it easier for administrators to access user data

☒ To ensure that user passwords are protected from unauthorized access

☐ None of the above

Solution description

The primary purpose of securely storing user credentials (such as passwords) in a database is to protect them from unauthorized access and maintain the security and privacy of user accounts. Storing passwords securely involves techniques like hashing and salting to make it difficult for malicious actors to access and misuse the stored passwords. Options a and b are not the primary purposes and are not recommended practices for user credential storage.

(3) Library Application I

← Classroom

User Persistence & JWT Authentication
Library Application I

Problem Submissions Doubts

Library Application I

Easy • Score 20/20

Problem statement

[Send feedback](#)

In a library application, admins can add books to the system, and students can issue the available books. The application uses HTTP-basic authentication and persists user credentials. Choose the correct implementation of the `SecurityFilterChain` bean if we want to publicise the `/student/register` API.

Options: Pick one correct answer from below

☐

```
***
@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws
    Exception {
    http
        .csrf().disable()
        .authorizeHttpRequests()
        .antMatchers("/student/**").permitAll()
        .anyRequest()
        .authenticated()
        .and()
        .httpBasic();
    return http.build();
}
```

☒

```
***
@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws
    Exception {
    http
        .csrf().disable()
        .authorizeHttpRequests()
        .antMatchers("/student/register").permitAll()
        .anyRequest()
        .authenticated()
        .and()
        .httpBasic();
    return http.build();
}
```

☐

```
***
@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws
    Exception {
    http
        .csrf().disable()
        .authorizeHttpRequests()
        .antMatchers("/**").permitAll()
        .antMatchers("/student/register").hasRole("ALL")
        .anyRequest()
        .authenticated()
        .and()
        .httpBasic();
    return http.build();
}
```

☐ None of the above

(4) Library Application II

Problem Submissions Doubts

Library Application II

Easy • Score 20/20

Problem statement

[Send feedback](#)

In the Library Application, based on the given class for `User`, `Role` and `UserDTO`. Choose the correct service layer implementation for the student registration API `/student/register`, where you save the user as a student. (Note: In the database, we save rolesName as `"ROLESTUDENT"`, `ROLEADMIN"`).

User Entity:

```
***
@Entity
@Getter
@Setter
@ToString
@EqualsAndHashCode
public class User implements UserDetails {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String email;
    private String password;

    @ManyToMany(cascade = CascadeType.ALL, fetch = FetchType.EAGER)
    @JoinTable(name = "user_role", joinColumns = @JoinColumn(name = "user", referencedColumnName = "id"),
        inverseJoinColumns = @JoinColumn(name = "role", referencedColumnName = "id"))
    private Set<Role> roles = new HashSet<>();

    // setters, getters and overridden methods
}
```

Role Entity:

```
***
```

Options: Pick one correct answer from below

☐ A)

☒ B)

☐ C)

☐ D)

Solution description

```
***
public void createStudent(UserDTO userDto) {
    // Encoding the password using BCryptPasswordEncoder
    BCryptPasswordEncoder encoder = new BCryptPasswordEncoder();
    String encodedPassword = encoder.encode(userDto.getPassword());
    // Creating user object and setting email and password
    User user = new User();
    user.setEmail(userDto.getEmail());
    user.setPassword(encodedPassword);
    // Creating a role set and adding a role "ROLE_STUDENT" in it.
    Role role = new Role();
    Set<Role> roles = new HashSet<>();
    role.setRoleName("ROLE_STUDENT");
    roles.add(role);
    // Setting the role in the user object and saving it.
    user.setRoles(roles);
    userRepository.save(user);
}
```

Option B is correct, and the description is given in the image below

Role Entity:

```
@Entity
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Table(name = "role")
public class Role {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String roleName; //Student and Admin
}
```

UserRole:

```
@Data
public class UserRole {
    private String email;
    private String password;
}
```

A)

```
public void createStudent(UserDto userDto) {
    BCryptPasswordEncoder encoder = new BCryptPasswordEncoder();
    String encodedPassword = encoder.encode(userDto.getPassword());
    User user = new User();
    user.setEmail(userDto.getEmail());
    user.setPassword(encodedPassword);
    user.setRoles("ROLE_STUDENT");
    userRepository.save(user);
}
```

B)

```
public void createStudent(UserDto userDto) {
    BCryptPasswordEncoder encoder = new BCryptPasswordEncoder();
    String encodedPassword = encoder.encode(userDto.getPassword());
    User user = new User();
    user.setEmail(userDto.getEmail());
    user.setPassword(encodedPassword);
    Role role = new Role();
    Set<Role> roles = new HashSet<>();
    role.setRoleName("ROLE_STUDENT");
    roles.add(role);
    user.setRoles(roles);
    userRepository.save(user);
}
```

C)

```
public void createStudent(UserDto userDto) {
    User user = new User();
    user.setEmail(userDto.getEmail());
    user.setPassword(userDto.getPassword());
    Role role = new Role();
    Set<Role> roles = new HashSet<>();
    role.setRoleName("ROLE_STUDENT");
    roles.add(role);
    user.setRoles(roles);
    userRepository.save(user);
}
```

(5) Purpose of Remember Me

← Classroom

User Persistence & JWT Authentication

Purpose of Remember Me

Problem Submissions Doubts

✓ Purpose of Remember Me

Easy • Score 20/20

Send feedback

What is the purpose of the "Remember Me" feature in session management?

Options: Pick one correct answer from below

Attempts left: 0/2

☐ To remember the user's username so that he/she doesn't have to enter again.

☐ To keep the user's session active indefinitely.

☒ To provide a convenient way for users to log in quickly. ✓

☐ To lock the user out of their account after inactivity.

Solution description

The "Remember Me" feature in session management is designed to allow users to log in more conveniently and quickly without having to enter their credentials (username and password) every time they access a web application. This feature is typically implemented by extending the duration of the user's session, often with a long-lasting cookie so that they remain logged in across multiple sessions or visits.

(6) ThymLeaf

← Classroom

User Persistence & JWT Authentication
ThymLeaf

?

V

Problem Submissions Doubts

ThymLeaf

Easy • Score 20/20

Problem statement

What is Thymeleaf?

Send feedback

Options: Pick one correct answer from below

Attempts left: 1/2

☐ A JavaScript framework for building web applications.

☐ An object-relational mapping (ORM) framework.

☒ A template engine for server-side Java applications.

☐ A JSP which allows the creation of front-end web pages.

Solution description

Thymeleaf is a popular Java-based template engine for creating dynamic web pages in server-side Java applications. It allows developers to define templates with HTML that can be dynamically populated with data from the server side using Java or other back-end technologies. Thymeleaf is widely used in Java web development and is not a server-side programming language, an ORM framework, or a JSP (JavaServer Pages).

(7) Default Session Cookie

← Classroom

User Persistence & JWT Authentication
Default Session Cookie

?

V

Problem Submissions Doubts

Default Session Cookie

Easy • Score 20/20

Problem statement

What is the default session cookie name in Java-based web applications?

Send feedback

Options: Pick one correct answer from below

Attempts left: 1/2

☐ SESSIONID

☐ COOKIE_SESSION

☒ JSESSIONID

☐ SERVER_COOKIE

Solution description

This is a common default session cookie name used by Java-based web applications, including Servlets and Java EE. The "JSESSIONID" cookie tracks the user's session on the server.

(8) Library Application III

Problem Submissions Doubts

Library Application III

Easy • Score 20/20

Problem statement

In the library application, we must create a login page for user login functionality using thymeleaf. The login API is "/login". Choose the correct implementation of the login page.

Send feedback

Options: Pick one correct answer from below

☐ A)

☐ B)

☒ C)

☐ D)

A)

```
<!--THYME HTML-->
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Login</title>
</head>
<body>
<div class="container-fluid text-center">
<form href="/login" method="post" style="max-width: 350px; margin: 0 auto;">
<div class="border border-secondary p-3 rounded">
<div>Enter Login Credential</div>
<input type="text" name="username" class="form-control"
placeholder="Username" required autofocus/>
</div>
<input type="password" name="password" class="form-control"
placeholder="Password" required />
</div>
<input type="checkbox" name="remember-me" /> <small>Remember Me</small>
</div>
<input type="submit" value="Login" class="btn btn-primary" />
</div>
</form>
</div>
</body>
</html>
```

```

<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0,
minimum-scale=1.0">
<title>Login</title>
</head>
<body>
<div class="container-fluid text-center">
<form th:action="@{/login}" method="post" style="max-width: 350px; margin: 0 auto;">
<div class="border border-secondary p-3 rounded">
<p>Enter Login Credential</p>
<p><input type="text" name="username" class="form-control"
placeholder="Username" required autofocus/>
</p>
<p><input type="password" name="password" class="form-control"
placeholder="Password" required />
</p>
<p><input type="checkbox" name="remember-me" />&nbsp;&nbsp;&nbsp;Remember Me
</p>
<p><input type="submit" value="Login" class="btn btn-primary" />
</p>
</div>
</form>
</div>
</body>
</html>

```

C)

```

<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0,
minimum-scale=1.0">
<title>Login</title>
</head>
<body>
<div class="container-fluid text-center">
<form th:action="@{/login}" method="post" style="max-width: 350px; margin: 0 auto;">
<div class="border border-secondary p-3 rounded">
<p>Enter Login Credential</p>
<p><input type="text" name="username" class="form-control"
placeholder="Username" required autofocus/>
</p>
<p><input type="password" name="password" class="form-control"
placeholder="Password" required />
</p>
<p><input type="checkbox" name="remember-me" />&nbsp;&nbsp;&nbsp;Remember Me
</p>
<p><input type="submit" value="Login" class="btn btn-primary" />
</p>
</div>
</form>
</div>
</body>
</html>

```

D) None of the above

(9) Library Application IV

Problem

Submissions

Doubts

Send feedback

Problem statement

In the library application, we must implement the remember-me functionality. We have a login.html page for user login. The login API is "/login". Choose the correct implementation of the **SecurityFilterChain** bean for the same.

A)

```

@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
    http.csrf().disable()
        .authorizeHttpRequests()
        .antMatchers("/student/register").permitAll()
        .and()
        .rememberMe().userDetailsService(userDetailsService)
        .and()
        .httpBasic()
        .loginPage("/login")
        .permitAll()
        .and()
        .logout().deleteCookies("remember-me");
    return http.build();
}

```

B)

```

@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
    http.csrf().disable()
        .authorizeHttpRequests()
        .antMatchers("/student/register").permitAll()
        .and()
        .rememberMe().userDetailsService(userDetailsService)
        .and()
        .formLogin()
        .loginPage("/login")
        .allowAll()
        .and()
        .logout().deleteCookies("remember-me");
    return http.build();
}

```

Options: Pick one correct answer from below

☐ A)

☐ B)

☒ C)

☐ D)

Solution description

```

@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
    http.csrf().disable()
        .authorizeHttpRequests()
        .antMatchers("/student/register").permitAll()
        .and()
        // Configuration to remember userDetailsService
        .rememberMe().userDetailsService(userDetailsService)
        .and()
        // Enabling form based login and making the "/login" API public
        .formLogin()
        .loginPage("/login")
        .permitAll()
        .and()
        // Delete the cookie after logout
        .logout().deleteCookies("remember-me");
    return http.build();
}

```

Option C is correct, and the description is given in the image below

(10) JWT Signature

← Classroom

User Persistence & JWT Authentication
JWT Signatures

Problem Submissions Doubts

✓ JWT Signature

Easy • Score 20/20

Problem statement

What is the purpose of the JWT signature?

Send feedback

Options: Pick one correct answer from below

☐ To make the token human-readable.

☐ To provide data for the application.

☒ To verify the integrity and authenticity of the token.

☐ To store metadata about the token.

Solution description

The JWT signature ensures the token is authentic and has not been tampered with. It allows the recipient to verify that the token has not been modified during transit and was indeed issued by a trusted entity.

(11) Parts of JWT Token

← Classroom

User Persistence & JWT Authentication
Parts of JWT Token

Problem Submissions Doubts

✓ Parts of JWT Token

Easy • Score 20/20

Problem statement

What are the three main parts of a JSON Web Token (JWT)?

Send feedback

Options: Pick one correct answer from below

Attempts left: 1/2

☐ Payload, Data, Key

☐ Header, Data, Authentication

☐ Token, Claims, Signature

☒ Header, Body, Signature

Solution description

In a JSON Web Token (JWT), there are three main parts:
1. Header: The header contains metadata about the token, such as the type of token (JWT) and the signing algorithm used. It is encoded as a JSON object.
2. Body (Payload): The payload contains claims or data. Claims are statements about an entity (typically, the user) and additional data. There are three types of claims: registered, public, and private claims. The payload is also encoded as a JSON object.
3. Signature: The signature is a cryptographic signature created by taking the encoded header, encoded payload, a secret key, and the specified algorithm. It is used to verify the token's integrity and authenticity.

(12) Claims

← Classroom

User Persistence & JWT Authentication
Claims

Problem Submissions Doubts

✓ Claims

Easy • Score 20/20

Problem statement

What are claims in the context of JWT authentication?

Send feedback

Options: Pick one correct answer from below

Attempts left: 1/2

☒ Claims are statements about an entity and additional data.

☐ Claims are cryptographic keys used for signing JWTs.

☐ Claims are reserved keywords used for encoding the header.

☐ Claims are encryption algorithms used to secure JWTs.

Solution description

In JWT (JSON Web Tokens) authentication, claims are information or statements about an entity (typically, a user) and any additional data relevant to the authentication process. These claims convey user roles, permissions, and other attributes about the token's subject.

(13) Once Per Request Filter

← Classroom

User Persistence & JWT Authentication
OncePerRequestFilter

?

V

Problem Submissions Doubts

OncePerRequestFilter

Easy • Score 20/20

Send feedback

Problem statement

What is the primary purpose of the OncePerRequestFilter in Spring Security?

Options: Pick one correct answer from below

Attempts left: 1/2

☐ To filter requests only once per session.

☒ To execute filtering logic for every incoming request.

☐ To maintain the session state for users.

☐ To filter requests based on the request method.

Solution description

The primary purpose of the OncePerRequestFilter in Spring Security is to execute filtering logic for every incoming HTTP request. It ensures the filtering logic is applied once per request, regardless of the user's session or other factors. This is important for performing custom security checks and modifications on each request.

(14) Once Per Request Filter Usage

← Classroom

User Persistence & JWT Authentication
OnePerRequestFilter Usage

?

V

Problem Submissions Doubts

OnePerRequestFilter Usage

Easy • Score 20/20

Send feedback

Problem statement

In Spring Security, when is the OncePerRequestFilter typically used?

Options: Pick one correct answer from below

Attempts left: 1/2

☐ To handle user authentication and authorization.

☐ To manage user sessions and cookies.

☒ To customize and add custom filtering logic.

☐ To create custom request mappings.

Solution description

The OncePerRequestFilter is typically used in Spring Security when you need to customize and add custom filtering logic to the filter chain. It allows you to perform request parameter validation, custom headers, or other security-related checks on each incoming request.

(15) Session While Using JWT

← Classroom

User Persistence & JWT Authentication
Session while using JWT

?

V

Problem Submissions Doubts

Session while using JWT

Easy • Score 20/20

Send feedback

Problem statement

In the context of JWT authentication, what is a characteristic of the session management approach?

Options: Pick one correct answer from below

Attempts left: 1/2

☐ Sessions are stored on the server, and tokens are used only for identity.

☒ Stateless sessions are used, eliminating the need for server-side session storage.

☐ Tokens are constantly refreshed, leading to increased server load.

☐ Users' session data is stored within the JWT token itself.

Solution description

In the context of JWT authentication, one of the significant advantages is that it allows for stateless sessions. JWTs contain all the necessary information to identify and authenticate a user, eliminating the need for server-side session storage, which is a characteristic of stateless authentication.

(16) Token Duration

Classroom

User Persistence & JWT Authentication
Library Application V

Problem Submissions Doubts

Token duration

Easy Score 20/20

Send feedback

Problem statement

Continuing the library application, we need to implement JWT authentication. Choose the correct implementation of the generateToken() method for creating a JWT token with a 25-minute expiration time.

A)

```
***
private String secret = "thisIsTheSecretKeyForLibraryApplication";
private static final long JWT_TOKEN_VALIDITY = 15000;
public String generateToken(UserDetails userDetails) {
    MapString, Object> claims = new HashMap<>();
    return Jwts.builder().setClaims(claims).setSubject(userDetails.getUsername()).setIssuedAt(new Date(System.currentTimeMillis())).setExpiration(new Date(System.currentTimeMillis() + JWT_TOKEN_VALIDITY * 1000)).signWith(SignatureAlgorithm.HS256, getSecret()).compact();
}
```

B)

```
***
private String secret = "thisIsTheSecretKeyForLibraryApplication";
private static final long JWT_TOKEN_VALIDITY = 1500;
public String generateToken(UserDetails userDetails) {
    MapString, Object> claims = new HashMap<>();
    return Jwts.builder().setClaims(claims).setSubject(userDetails.getUsername()).setIssuedAt(new Date(System.currentTimeMillis())).setExpiration(new Date(System.currentTimeMillis() + JWT_TOKEN_VALIDITY * 1000)).signWith(SignatureAlgorithm.HS256, getSecret()).compact();
}
```

C)

```
***
private String secret = "thisIsTheSecretKeyForLibraryApplication";
private static final long JWT_TOKEN_VALIDITY = 150;
public String generateToken(UserDetails userDetails) {
    MapString, Object> claims = new HashMap<>();
    return Jwts.builder().setClaims(claims).setSubject(userDetails.getUsername()).setIssuedAt(new Date(System.currentTimeMillis())).setExpiration(new Date(System.currentTimeMillis() + JWT_TOKEN_VALIDITY * 1000)).signWith(SignatureAlgorithm.HS256, getSecret()).compact();
}
```

D) None of the above

Options: Pick one correct answer from below

☒ A)

☐ B)

☐ C)

☐ D)

Solution description

Option A is correct, and the description is given in the image below

```
***
private String secret = "thisIsTheSecretKeyForLibraryApplication";
private static final long JWT_TOKEN_VALIDITY = 15000;
public String generateToken(UserDetails userDetails) {
    MapString, Object> claims = new HashMap<>();
    return Jwts.builder().setClaims(claims).setSubject(userDetails.getUsername()).setIssuedAt(new Date(System.currentTimeMillis())).setExpiration(new Date(System.currentTimeMillis() + JWT_TOKEN_VALIDITY * 1000)).signWith(SignatureAlgorithm.HS256, getSecret()).compact();
}
```

(17) Library Application V

Classroom

User Persistence & JWT Authentication
Library Application VI

Problem Submissions Doubts

Library Application V

Easy Score 20/20

Send feedback

Problem statement

Continuing the library application, we have now implemented JWT authentication. For its configuration, we need to create SecurityFilterChain bean with STATELESS session creation policy. Also we need to make `/student/register`, `/login` as public APIs and, `/admin/**` as admin APIs and `/student/**` as student APIs. Choose the correct implementation of the SecurityFilterChain bean.

A)

```
***
@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception
{
    http
        .csrf().disable()
        .authorizeHttpRequests()
        .antMatchers("/student/register", "/login").permitAll()
        .antMatchers("/student/**").hasRole("ADMIN")
        .antMatchers("/admin/**").hasRole("STUDENT")
        .anyRequest()
        .authenticated()
        .and()
        .sessionManagement(session -> session.sessionCreationPolicy(STATELESS));
    http.addFilterBefore(filter, UsernamePasswordAuthenticationFilter.class);
    return http.build();
}
```

B)

```
***
@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception
{
    http
        .csrf().disable()
        .authorizeHttpRequests()
        .antMatchers("/student/register", "/auth/login").permitAll()
        .antMatchers("/student/**").hasRole("STUDENT")
        .antMatchers("/admin/**").hasRole("ADMIN")
        .anyRequest()
        .authenticated()
        .and()
        .sessionManagement(session -> session.sessionCreationPolicy(sessionCreationPolicy.STATELESS));
    http.addFilterBefore(filter, UsernamePasswordAuthenticationFilter.class);
    return http.build();
}
```

Options: Pick one correct answer from below

☐ A)

☐ B)

☒ C)

☐ None of the above

Solution description

Option C is correct, and the description is given in the image below

```
***
@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception
{
    http
        .csrf().disable()
        .authorizeHttpRequests()
        // making registration and login APIs public
        .antMatchers("/student/register", "/login").permitAll()
        // Making APIs starting with "/student" only accessible by user with "STUDENT" role
        .antMatchers("/student/**").hasRole("STUDENT")
        // Making APIs starting with "/admin" only accessible by user with "ADMIN" role
        .antMatchers("/admin/**").hasRole("ADMIN")
        .anyRequest()
        .authenticated()
        .and()
        // Setting "STATELESS" session creation policy
        .sessionManagement(session -> session.sessionCreationPolicy(sessionCreationPolicy.STATELESS));
    http.addFilterBefore(filter, UsernamePasswordAuthenticationFilter.class);
    return http.build();
}
```