SecureTokenServiceImplTest Documentation

SecureTokenServiceImplTest Documentation

Overview

This document outlines the unit tests designed for the SecureTokenServiceImpl class in our authentication module.

The service manages secure tokens for user validation and account operations (e.g., password resets, email verification).

It interacts with SecureTokenRepository and UserRepository.

Test Class Structure

Dependencies

The test class uses:

- JUnit 5 - for structuring and running tests

- Mockito - for mocking dependencies

- Spring Boot Test - for integration within the Spring context

Mocked Components

- SecureTokenRepository - to simulate secure token persistence

- UserRepository - to simulate user data access

Test Setup

Each test initializes:

1. Mocked repositories via MockitoAnnotations.openMocks(this)

2. Sample test data:

- A User object with a randomly generated UUID

- A SecureToken instance associated with the user

Test Cases

1. testCreateToken_SuccessfulSave

Purpose: Verify that a secure token is saved correctly through the repository.

Scenario:

- Arrange: Prepare a test token and mock tokenRepo.save(...) to return it

- Act: Call createToken(token)

- Assert: Confirm the saved token is returned and the repo method is invoked

Business Logic Verified:

- The service properly delegates token saving to the repository

2. testGetToken_ValidToken_ReturnsToken

Purpose: Ensure a valid (non-expired) token is retrieved correctly.

Scenario:

- Arrange: Create a token with a future expiration date and mock findByToken(...)

- Act: Call getToken(tokenStr)

- Assert: Confirm the correct token is returned

Business Logic Verified:

- Token filtering excludes expired tokens

- Valid tokens are passed through

3. testGetToken_ExpiredToken_RemovedAndReturnsEmpty

Purpose: Ensure expired tokens are removed and not returned.

Scenario:

- Arrange: Create an expired token, and mock findByToken(...) to return it

- Act: Call getToken(tokenStr)

- Assert: Confirm empty Optional is returned and removeByToken(...) is called

Business Logic Verified:

- Token expiration logic is enforced

- Expired tokens are proactively cleaned up

4. testRemoveToken_CallsRepository

Purpose: Verify that calling removeToken() triggers token deletion.

Scenario:

- Arrange: Mock removeByToken(...)

- Act: Call removeToken(tokenStr)

- Assert: Verify that removeByToken() was called once with the correct token

Business Logic Verified:

- Service properly delegates token removal

5. testGenerateForUser_ValidUser_GeneratesAndSavesToken

Purpose: Ensure a secure token is generated for a valid user, and that it is associated correctly.

Scenario:

- Arrange: Mock userRepo.findById() to return a test user

- Act: Call generateForUser(userId, 30)

- Assert: Verify:

  - Token is generated with proper format and duration

- User's token field is updated
- Both userRepo.save() and tokenRepo.save() are invoked

Business Logic Verified:

- Tokens are generated with UUID and expiration timestamp
- Tokens are correctly linked to users and saved

Mocking Strategy

- Token lookup: simulate token repository queries with expiration logic
- User loading: mock user fetch and verify behavior on invalid IDs
- Save/remove: mock write operations to repositories

Test Coverage

Covers all major functionality of SecureTokenServiceImpl:

- Saving tokens
- Filtering expired tokens
- Removing tokens
- Generating new tokens for users

Conclusion

These tests validate the critical functionality of SecureTokenServiceImpl for secure user token operations.

They ensure correct repository interactions, token lifecycle handling, and robust user association logic.

As the authentication system grows, these tests will help maintain the reliability of token-based operations.