SecureTokenServiceTest Documentation

SecureTokenServiceTest Documentation

Overview:

This document outlines the unit tests implemented for the SecureTokenService interface, tested through

its concrete implementation SecureTokenServiceImpl. The service manages creation, validation, and deletion

of secure tokens linked to users, often used for email verification and password reset functionality.

Test Class Structure:

Dependencies:

- JUnit 5 for test execution and assertions
- Mockito for mocking dependencies
- Spring Boot Test for integrating with the Spring context

Mocked Components:

- SecureTokenRepository: Simulates persistence and lookup of tokens
- UserRepository: Simulates user lookup and saving

Test Setup:

- Initializes mocks via MockitoAnnotations
- Creates a test User and SecureToken
- Injects mocks into a SecureTokenServiceImpl instance

Test Cases:

1. testCreateToken_SuccessfulSave

Purpose: Ensure that a token can be saved via the repository.

Scenario:

- Mocks save behavior of tokenRepo

- Asserts the returned token is not null and correctly saved

- Verifies save was called once

2. testGetToken_ValidToken_ReturnsToken

Purpose: Validate that a non-expired token is retrievable.

Scenario:

- Mocks findByToken to return a valid token

- Asserts that token is returned and matches expected values

3. testGetToken_ExpiredToken_RemovedAndReturnsEmpty

Purpose: Ensure expired tokens are deleted and not returned.

Scenario:

- Sets token expiration to a past timestamp

- Asserts result is empty and removeByToken was called

4. testRemoveToken_CallsRepository

Purpose: Verify removal method delegates to repository.

Scenario:

- Mocks removeByToken call

- Verifies repository method was invoked once

5. testGenerateForUser_ValidUser_GeneratesAndSavesToken

Purpose: Generate a token, link it to a user, and save both.

Scenario:

- Mocks userRepo.findById to return a test user
- Mocks tokenRepo.save and userRepo.save
- Asserts token has valid structure and expiration
- Verifies save operations on both repositories

Mocking Strategy:

- All repositories are mocked to simulate interactions and isolate logic
- Save and find behaviors are predefined using Mockito
- Verification is used to ensure repository methods are triggered

Test Coverage:

- Token creation, retrieval, expiration filtering, and removal
- Generation and user-token linkage
- Repository behavior validation

Conclusion:

These unit tests thoroughly validate the SecureTokenServiceImpl implementation of the SecureTokenService
interface. They ensure business logic correctness and confirm that tokens are managed securely and reliably.