CategoryServiceAdvancedTest Documentation

Overview

This document describes additional unit tests for the CategoryService class. These tests cover edge cases and error handling not included in the original test suite. The CategoryService handles CRUD operations for product categories in the e-commerce system.

Test Class Structure

Dependencies

- JUnit 5: For writing and executing unit tests
- Mockito: For mocking repository behavior and isolating service logic

Mocked Components

- CategoryRepository: Interface for accessing and modifying category data in the database

Test Setup

1. Initialize mocks using MockitoAnnotations
2. Create a sample testCategory object
3. Inject mocks into the CategoryService

Test Cases

1. testAddCategory_NullName

Purpose: Ensure the service properly rejects null category names.

Test Scenario:

- Arrange:
  Call addCategory with null as the category name.

- Act:
  Invoke categoryService.addCategory(testCategory, null).

- Assert:
  Verify that the response returns HTTP 400 and no save operation is triggered.

Business Logic Verified:

The service must prevent saving categories with null or empty names.

2. testFindById_CategoryFound

Purpose: Ensure a category is returned successfully when a valid ID is provided.

Test Scenario:

- Arrange:
  Mock repository to return a valid category on findById.

- Act:
  Call categoryService.findById with a known valid ID.

- Assert:
  Assert that the returned category is not null and matches expected values.

Business Logic Verified:

Valid category IDs should retrieve the associated category from the system.

3. testDeleteCategory_CategoryNotFound

Purpose: Ensure the service throws an exception when deleting a non-existent category.

Test Scenario:

- Arrange:
  Mock existsById to return false for a fake ID.

- Act:
  Call categoryService.deleteCategory with a fake ID.

- Assert:
  Assert that NoSuchElementException is thrown and deleteById is never called.

Business Logic Verified:

Deleting a non-existent category must be safely handled with clear error signaling.

Mocking Strategy

Mocking is used to simulate repository behavior without requiring a real database:

- Verify method calls and arguments
- Simulate repository responses like Optional.empty or valid returns
- Ensure the service handles both success and failure cases

Conclusion

These tests strengthen confidence in the CategoryService's robustness, especially when facing invalid input or missing data. They ensure the application provides clear and reliable behavior in all conditions.