

Tests unitaires

On utilise souvent la matrice **Given-When-Then**.

On isole le service à tester en utilisant des Mock objects (objets factices).

- `@Mock` = Crée un objet factice (stub/mock)
 - Les repositories ne font rien de réel
 - leurs réponses sont contrôlées avec `when(...).thenReturn(...)`
- `@InjectMocks` = Crée la vraie instance à tester

Vérification du Workflow métier :

- `verify()` : "Le service a-t-il bien appelé cette méthode ?"
- `verifyNoInteractions()` : "Le service n'a-t-il PAS touché ce repository ?"

Exemple :

Si la catégorie n'existe pas → le service ne doit PAS essayer de sauver le produit !

Exemple de test

```
@ExtendWith(MockitoExtension::class)
class ProductServiceTest {

    @Mock
    private lateinit var productRepository: ProductRepository
    @Mock
    private lateinit var categoryRepository: CategoryRepository
    @InjectMocks
    private lateinit var productService: ProductService

    @Test
    @DisplayName("Should create product successfully when category exists")
    fun `should create product when category exists`() {
        // Given
        val categoryId = UUID.fromString("550e8400-e29b-41d4-a716-446655440010")
        val productId = UUID.fromString("550e8400-e29b-41d4-a716-446655440020")
        val createDTO = CreateProductDTO(
            name = "iPhone 15",
            description = "Latest Apple smartphone",
            price = BigDecimal("1199.99"),
            stock = 50,
            categoryId = categoryId
        )
        val category = Category(categoryId, "Smartphones", "Mobile devices")
        val savedProduct = Product(productId, "iPhone 15", "Latest Apple
        smartphone",
            BigDecimal("1199.99"), 50, category, ProductStatus.ACTIVE)
        `when`(categoryRepository.findById(categoryId)).thenReturn(Optional.of(category))
        `when`(productRepository.save(any(Product::class.java))).thenReturn(savedProduct)
```

```
// When
val result = productService.createProduct(createdDT0)

// Then
assertThat(result).isNotNull
assertThat(result.id).isEqualTo(productId)
assertThat(result.name).isEqualTo("iPhone 15")
assertThat(result.price).isEqualByComparingTo(BigDecimal("1199.99"))
assertThat(result.stock).isEqualTo(50)
verify(categoryRepository).findById(categoryId)
verify(productRepository).save(any(Product::class.java))
}

@Test
@DisplayName("Should throw exception when category not found")
fun `should throw exception when category not found`() {
    // Given
    val categoryId = UUID.fromString("550e8400-e29b-41d4-a716-446655440099")
    val createdDT0 = CreateProductDT0("iPhone 15", "Description",
BigDecimal("1199.99"), 50, categoryId)
    `when`(categoryRepository.findById(categoryId)).thenReturn(Optional.empty())

    // When & Then
    assertThatThrownBy { productService.createProduct(createdDT0) }
        .isInstanceOf(CategoryNotFoundException::class.java)
        .hasMessageContaining(categoryId.toString())
    verify(categoryRepository).findById(categoryId)
    verifyNoInteractions(productRepository)
}
}
```

From:
<http://slamwiki2.kobject.net/> - **SlamWiki 2.1**

Permanent link:
http://slamwiki2.kobject.net/eadl/bloc3/dev_av/tests/unit?rev=1758063738

Last update: **2025/09/17 01:02**

