

Project Mockito

In this project Mockito we will use the Mockito dependency to see how it helps us develop unit tests, and allows us to simulate the behavior of objects in isolation, without needing the actual implementation (such as database access, network calls, etc.).

First of all you need to add two dependencies to your pom.xml which are Spring Boot Test for unit tests and integration tests, and also Mockito, as we said to mock objects in tests.

```
<dependencies>
  <!-- Spring Boot Test for unit tests and integration tests -->
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>

  <!-- Mockito for mocking objects in tests -->
  <dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-core</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>
```

Next under src/test/java/com/example/gadgetapi/service we developed some tests to test the service:

```
package com.example.gadgetstore.service;
```

```
import com.example.gadgetstore.model.Gadget;
import com.example.gadgetstore.repository.GadgetRepository;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.MockitoAnnotations;
```

```
import java.util.Arrays;
import java.util.List;
import java.util.Optional;
```

```
import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.mockito.Mockito.*;
```

```
class GadgetServiceTest {
```

```

@Mock
private GadgetRepository gadgetRepository;

@InjectMocks
private GadgetService gadgetService;

private Gadget gadget1;
private Gadget gadget2;

@BeforeEach
void setUp() {
    // Initialize Mockito annotations
    MockitoAnnotations.openMocks(this);

    // Create sample gadgets for testing
    gadget1 = new Gadget(1L, "Smartphone", "BrandX", 499.99, "A great smartphone");
    gadget2 = new Gadget(2L, "Laptop", "BrandY", 899.99, "A powerful laptop");
}

@Test
void testGetAllGadgets() {
    // Mock the behavior of gadgetRepository.findAll() to return a list of gadgets
    when(gadgetRepository.findAll()).thenReturn(Arrays.asList(gadget1, gadget2));

    // Call the service method
    List<Gadget> gadgets = gadgetService.getAllGadgets();

    // Verify the results
    assertEquals(2, gadgets.size());
    assertEquals("Smartphone", gadgets.get(0).getName());
    assertEquals("Laptop", gadgets.get(1).getName());

    // Verify that the findAll method in gadgetRepository was called once
    verify(gadgetRepository, times(1)).findAll();
}

@Test
void testGetGadgetById() {
    // Mock the behavior of gadgetRepository.findById()
    when(gadgetRepository.findById(1L)).thenReturn(Optional.of(gadget1));

    // Call the service method
    Optional<Gadget> foundGadget = gadgetService.getGadgetById(1L);

```

```
// Verify the results
assertEquals("Smartphone", foundGadget.get().getName());

// Verify that findById was called with the correct argument
verify(gadgetRepository, times(1)).findById(1L);
}
```

```
@Test
void testSaveGadget() {
    // Mock the behavior of gadgetRepository.save()
    when(gadgetRepository.save(gadget1)).thenReturn(gadget1);

    // Call the service method
    Gadget savedGadget = gadgetService.saveGadget(gadget1);

    // Verify the result
    assertEquals("Smartphone", savedGadget.getName());

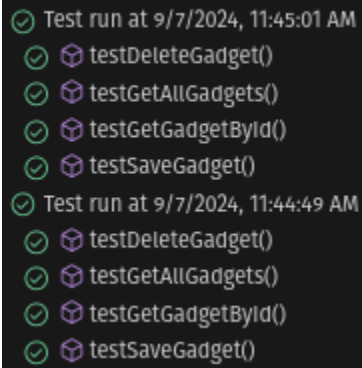
    // Verify that save was called
    verify(gadgetRepository, times(1)).save(gadget1);
}
```

```
@Test
void testDeleteGadget() {
    // Mock the behavior of gadgetRepository.deleteById()
    doNothing().when(gadgetRepository).deleteById(1L);

    // Call the service method
    gadgetService.deleteGadget(1L);

    // Verify that deleteById was called with the correct argument
    verify(gadgetRepository, times(1)).deleteById(1L);
}
}
```

And as you can see our IDE tells us they were all passed:



The screenshot shows a dark-themed IDE window with test results. It lists two test runs, each with four sub-tests, all of which passed. Each item is preceded by a green checkmark icon. The sub-tests are preceded by a purple icon that looks like a cube or a box.

- ✓ Test run at 9/7/2024, 11:45:01 AM
 - ✓ testDeleteGadget()
 - ✓ testGetAllGadgets()
 - ✓ testGetGadgetById()
 - ✓ testSaveGadget()
- ✓ Test run at 9/7/2024, 11:44:49 AM
 - ✓ testDeleteGadget()
 - ✓ testGetAllGadgets()
 - ✓ testGetGadgetById()
 - ✓ testSaveGadget()