JSE Testing

Integrate JUnit and Mockito, Unit Testing for Service Layer

* Unit tests are typically **automated tests** written and run by software developers to ensure that a section of an application (known as the “unit”) meets its design and behaves as intended

@ExtendWith(MockitoExtension.class)

class UserServiceTest {

@Mock

private UserRepository userRepository;

@InjectMocks

private UserService userService;

}

1. We create this class extend with MockitoExtension, so don’t need to declare MockitoAnnotations.initMocks(this) .
2. @Mock : because on UserService have dependency to UserRepository, we create mock object to simulate behaviour of the real object.
3. @InjectMock : is create Object to inject mock dependency, because on this case we simulate about UserService, so we create @InjectMock based on UserService.

Step 1 : create maven project

Step 2 : pom.xml

<properties>

<maven.compiler.plugin.version>3.8.4</maven.compiler.plugin.version>

<maven.compiler.target>11</maven.compiler.target>

<maven.compiler.source>11</maven.compiler.source>

<junit-jupiter.version>5.5.2</junit-jupiter.version>

</properties>

<dependencies>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>4.6.1</version>

<scope>test</scope>

</dependency>

<!-- junit 5 -->

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-engine</artifactId>

<version>${junit-jupiter.version}</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.junit.platform</groupId>

<artifactId>junit-platform-launcher</artifactId>

<version>1.5.1</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-junit-jupiter</artifactId>

<version>2.19.0</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-site-plugin</artifactId>

<version>3.7.1</version>

</plugin>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-project-info-reports-plugin</artifactId>

<version>3.0.0</version>

</plugin>

<!-- Need at least 2.22.0 to support JUnit 5 -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>3.0.0-M3</version>

</plugin>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.1</version>

<configuration>

<source>11</source>

<target>11</target>

</configuration>

</plugin>

</plugins>

</build>

<reporting>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-report-plugin</artifactId>

</plugin>

</plugins>

</reporting>

</project>

Step 3 : com.service

**public** **class** HelloService {

**public** String sayHello(String name) {

**return** "Welcome " + name;

}

}

Step 4: com.main

**public** **class** HelloWorld {

**public** **static** **void** main(String[] args) {

HelloWorld helloWorld = **new** HelloWorld();

String data = helloWorld.callService("Happy");

System.***out***.println(data);

}

**public** String callService(String name) {

HelloService helloService = **new** HelloService();

**return** helloService.sayHello(name);

}

}

Testing

Step 5 : src/test/java/com/main

**import** **static** org.junit.jupiter.api.Assertions.*assertEquals*;

**import** org.junit.jupiter.api.Test;

**import** org.junit.jupiter.api.extension.ExtendWith;

**import** org.mockito.InjectMocks;

**import** org.mockito.Mock;

**import** org.mockito.Mockito;

**import** org.mockito.junit.jupiter.MockitoExtension;

**import** org.mockito.junit.jupiter.MockitoSettings;

**import** org.mockito.quality.Strictness;

**import** com.service.HelloService;

@ExtendWith(MockitoExtension.**class**)

@MockitoSettings(strictness = Strictness.***LENIENT***)

//org.mockito.exceptions.misusing.UnnecessaryStubbingException: Unnecessary stubbings detected.

//JUnit 5 tests you can silence this exception using this annotation

**class** HelloWorldTest {

@Mock

HelloService helloService;

@InjectMocks

HelloWorld helloWorld;

@Test

**public** **void** testMain() {

Mockito.*when*(helloService.sayHello("testing")).thenReturn("Welcome testing");

// HelloWorld helloWorld = new HelloWorld();

String actual = helloWorld.callService("testing");

*assertEquals*("Welcome testing", actual);

}

}