使用 Mockito 在 Java 測試中

要在 Java 測試中使用 Mockito，請按照以下步驟創建模擬對象、模擬方法並驗證交互：

1. 添加依賴

在 `pom.xml` (Maven) 中包含 Mockito 和 JUnit：

```
<dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-core</artifactId>
    <version>5.3.1</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter</artifactId>
    <version>5.9.2</version>
    <scope>test</scope>
</dependency>
```

2. 設置測試類

使用 JUnit 5 和 Mockito 的擴展：

```
import org.junit.jupiter.api.extension.ExtendWith;
import org.mockito.junit.jupiter.MockitoExtension;

@ExtendWith(MockitoExtension.class)
public class UserServiceTest {
    // ...
}
```

3. 創建模擬和注入依賴

- 使用 `@Mock` 來模擬您想要模擬的依賴。
- 使用 `@InjectMocks` 將模擬對象注入到測試類中。

```
@Mock
private UserRepository userRepository;
```

```
@InjectMocks
private UserService userService; // UserService 依賴於 UserRepository
```

4. 模擬方法調用

使用 `when().thenReturn()` 來定義模擬行為：

```
@Test
void getUserById_ReturnsUser_WhenUserExists() {
    User mockUser = new User(1L, "Alice");
    // 模擬儲存庫的方法
    when(userRepository.findById(1L)).thenReturn(mockUser);

    User result = userService.getUserById(1L);

    assertEquals("Alice", result.getName());
}
```

5. 驗證交互

檢查模擬方法是否按預期調用：

```
@Test
void getUserById_CallsRepository() {
    userService.getUserById(1L);
    // 驗證儲存庫方法已被調用一次，ID 為 1
    verify(userRepository, times(1)).findById(1L);
}
```

常見用例

模擬異常

```
@Test
void getUserById_ThrowsException_WhenRepositoryFails() {
    when(userRepository.findById(anyLong())).thenThrow(new RuntimeException("DB Error"));

    assertThrows(RuntimeException.class, () -> userService.getUserById(1L));
}
```

參數匹配器 使用 `any()`, `eq()` 等來靈活匹配參數：

```
when(userRepository.findById(anyLong())).thenReturn(new User(1L, "Bob"));
```

BDD (行為驅動開發) 風格 使用 `given().willReturn()` 來提高可讀性：

```
import static org.mockito.BDDMockito.*;

@Test
void bddStyleExample() {
    given(userRepository.findById(1L)).willReturn(new User(1L, "Alice"));

    User result = userService.getUserById(1L);

    then(userRepository).should().findById(1L);
    assertEquals("Alice", result.getName());
}
```

參數捕獲器 捕獲參數以進行詳細斷言：

```
@Test
void saveUser_CapturesArgument() {
    ArgumentCaptor<User> userCaptor = ArgumentCaptor.forClass(User.class);

    userService.saveUser("Charlie");

    verify(userRepository).save(userCaptor.capture());
    assertEquals("Charlie", userCaptor.getValue().getName());
}
```

關鍵注意事項

- **初始化**：使用 `@ExtendWith(MockitoExtension.class)` (JUnit 5) 或在 `@BeforeEach` 中使用 `MockitoAnnotations.openMock`
- **靜態模擬**：使用 `mockito-inline` 依賴來模擬靜態方法/構造函數。
- **間諜**：使用 `@Spy` 來包裹真實對象（部分模擬）。

故障排除

- **空模擬**：確保模擬已初始化（使用 `@ExtendWith` 或 `openMocks()`）。
- **模擬錯誤**：在測試中調用方法之前模擬方法。

有關高級用法，請參閱 Mockito 文檔。