

Develop JUnit Tests in BDD Style using BBDDMockito class

By Ramesh Fadatare (Java Guides)

Behavior-Driven development (BDD)

BDD encourages writing tests in a natural, human-readable language that focuses on the behavior of the application.

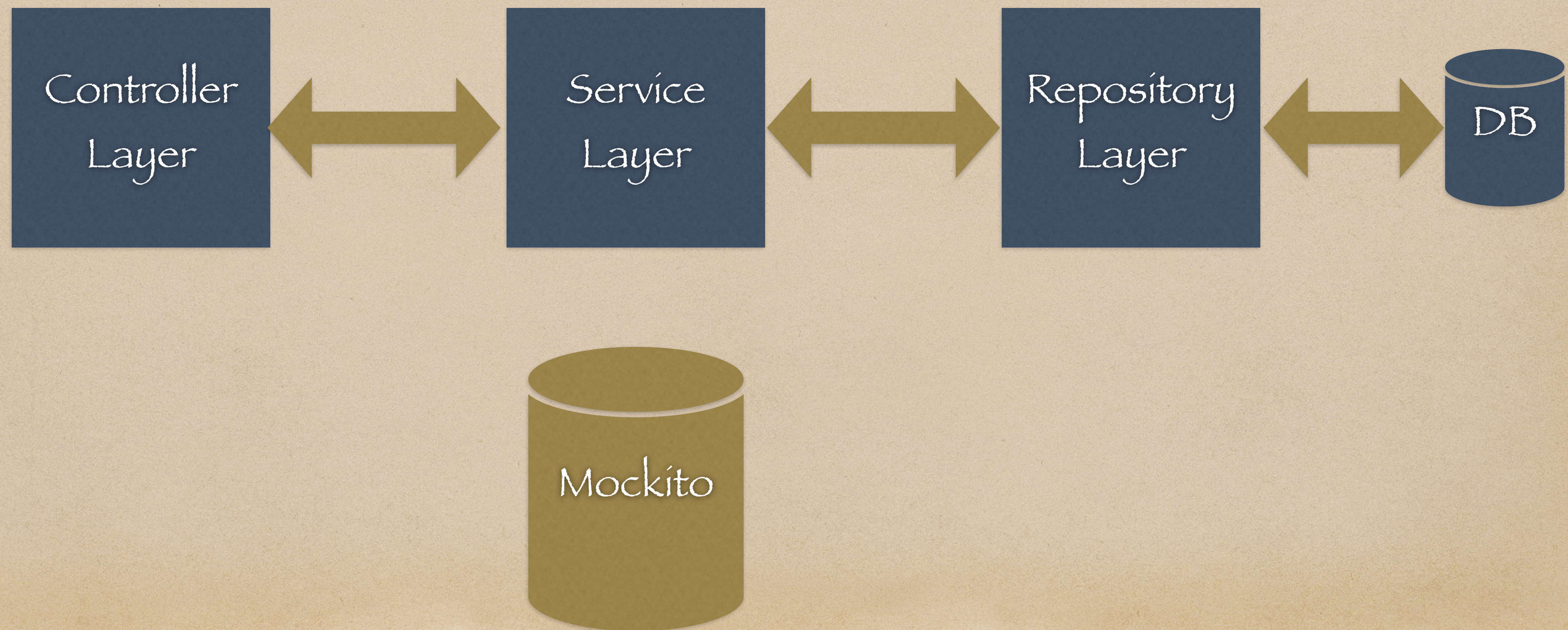
We write Unit Tests using a Behavior-Driven Development style (BDD) to increase the test readability (a lot).

It defines a clearly structured way of writing tests following three sections (Arrange, Act, Assert):

- **given** some preconditions (Arrange)
- **when** an action occurs (Act)
- **then** verify the output (Assert)

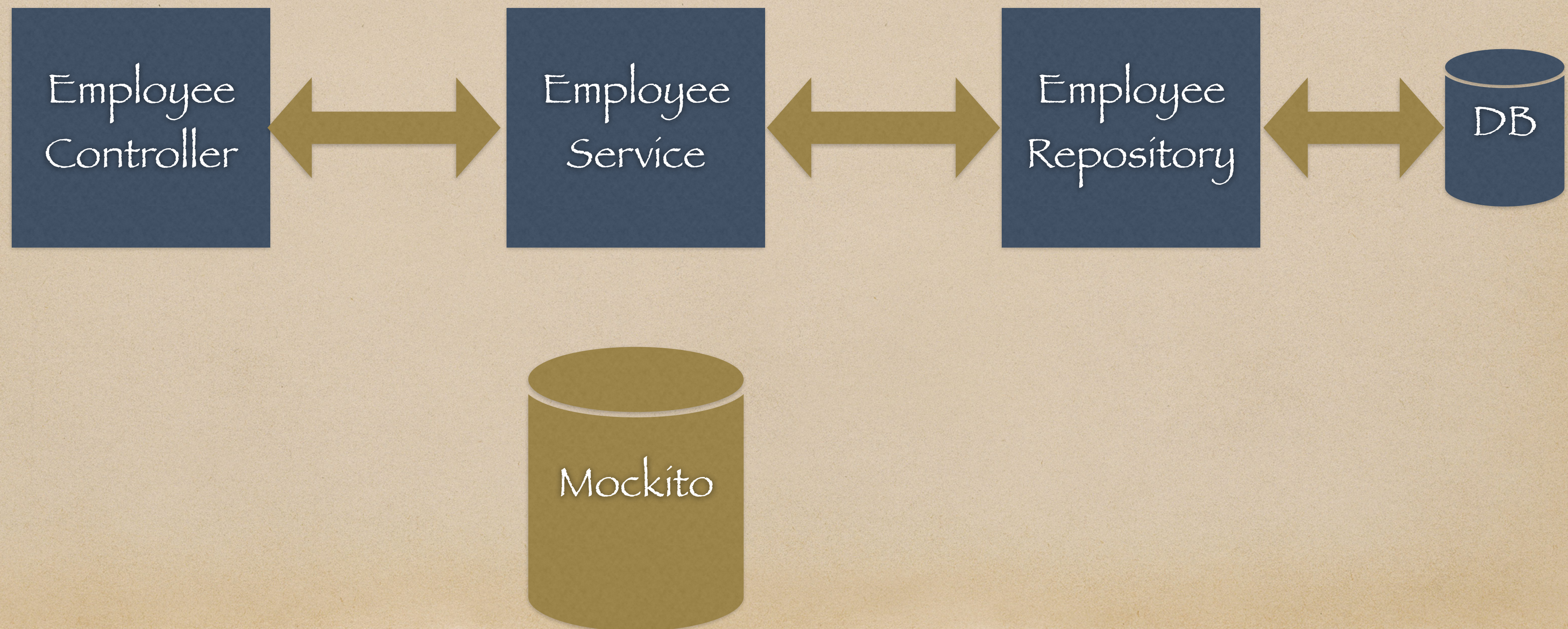
Spring Boot Application

Service Layer Testing



Spring Boot Application

Service Layer Testing



Mocking Dependencies using Mockito

Mockito mock() method - We can use **Mockito** class mock() method to create a mock object of a given class or interface. This is the simplest way to mock an object.

Mockito @Mock Annotation - We can mock an object using @Mock annotation too. It's useful when we want to use the mocked object at multiple places because we avoid calling mock() method multiple times. The code becomes more readable and we can specify mock object name that will be useful in case of errors.

Mockito @InjectMocks Annotation

When we want to inject a mocked object into another mocked object, we can use @InjectMocks annotation. @InjectMock creates the mock object of the class and injects the mocks that are marked with the annotations @Mock into it.

```
@ExtendWith(MockitoExtension.class)
public class EmployeeServiceTest {

    @Mock
    private EmployeeRepository employeeRepository;

    @InjectMocks
    private EmployeeServiceImpl employeeService;
```


BDDMockito Class

The Mockito library is shipped with a BDDMockito class which introduces BDD-friendly APIs.


Example: BDD style writing tests uses *//given* *//when* *//then* comments

```
@Test
public void givenEmployeesList_whenGetAllEmployees_thenGetEmployeesList(){
    // given (precondition)
    given(employeeRepository.findAll()).willReturn(List.of(employee, employee1));
    // when (action occurs)
    List<Employee> employees = employeeService.getAllEmployees();
    // then (verify the output)
    assertThat(employees.size()).isEqualTo(2);
}
```


Mockito vs. BDDMockito

The traditional mocking in Mockito is performed using **when(obj).thenReturn()** in the Arrange step.

```
// given  
Mockito.when(employeeRepository.findAll()).thenReturn(List.of(employee, employee1));
```



This BDDMockito allows us to take a more BDD friendly approach arranging our tests using **given().willReturn()**.

```
// given  
given(employeeRepository.findAll()).willReturn(List.of(employee, employee1));
```



Steps to use BDDMockito API


1. Add static import statement

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

2. Use given().willReturn() method



```
// given
```



```
given(employeeRepository.findAll()).willReturn(List.of(employee, employee1));
```