

[Home](#) › [Interview](#) › [Testing & Profiling/Sampling Java Apps Q&A](#) › [Unit Testing Q&A](#) › [JUnit Mockito Spring](#) › [Part 1: Unit testing with JUnit, Mockito & Spring](#)

# Part 1: Unit testing with JUnit, Mockito & Spring

Posted on [September 8, 2015](#) by [Arulkumaran Kumaraswamipillai](#)

**Step 1:** Create a simple Maven project from a command-line. Just press enter for all the prompts.

```
1 mvn archetype:generate -DgroupId=com.mytutorial -
```

**Step 2:** Import the above “simple-mocking-test” in the file system into eclipse. File → Import → “Existing Maven Projects” and then select the “simple-mocking-test” folder that was created by the “Step 1” from the file system.

**Step 3:** Add the testing, mocking, and Spring libraries to the pom.xml.

```
1 <project xmlns="http://maven.apache.org/POM/4.0.
2     xsi:schemaLocation="http://maven.apache.org/
3     <modelVersion>4.0.0</modelVersion>
4
5     <groupId>com.mytutorial</groupId>
6     <artifactId>simple-mocking-test</artifactId>
7     <version>1.0-SNAPSHOT</version>
8     <packaging>jar</packaging>
```

## 600+ Full Stack Java/JEE Interview Q&As ♥Free ♦FAQs

[open all](#) | [close all](#)

- ✚ Ice Breaker Interview
- ✚ Core Java Interview C
- ✚ JEE Interview Q&A (3
- ✚ Pressed for time? Jav
- ✚ SQL, XML, UML, JSC
- ✚ Hadoop & BigData Int
- ✚ Java Architecture Inte
- ✚ Scala Interview Q&As
- ✚ Spring, Hibernate, & I
  - ✚ Spring (18)
  - ✚ Hibernate (13)
  - ✚ AngularJS (2)
  - ✚ Git & SVN (6)
  - ✚ JMeter (2)
  - ✚ JSF (2)
  - ✚ Maven (3)
- ✚ Testing & Profiling/Sa
  - ✚ Automation Testing
  - ✚ Code Coverage (2)
  - ✚ Code Quality (2)
  - ✚ jvisualvm profiling (
  - ✚ Performance Testir

```

9
10 <name>simple-mocking-test</name>
11 <url>http://maven.apache.org</url>
12
13 <properties>
14     <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
15     <mockito-all.version>1.9.0</mockito-all.version>
16     <spring.version>3.2.13.RELEASE</spring.version>
17 </properties>
18
19 <dependencies>
20
21 <!-- JUnit Library -->
22 <dependency>
23     <groupId>junit</groupId>
24     <artifactId>junit</artifactId>
25     <version>4.11</version>
26     <scope>test</scope>
27 </dependency>
28
29 <!-- Mockito Library (for Mocking)-->
30 <dependency>
31     <groupId>org.mockito</groupId>
32     <artifactId>mockito-all</artifactId>
33     <version>${mockito-all.version}</version>
34     <scope>test</scope>
35 </dependency>
36
37 <!-- Dependency Injection Annotation @Na
38 <dependency>
39     <groupId>javax.inject</groupId>
40     <artifactId>javax.inject</artifactId>
41     <version>1</version>
42 </dependency>
43
44 <!-- Spring Dependencies (core libraries
45 <dependency>
46     <groupId>org.springframework</groupId>
47     <artifactId>spring-core</artifactId>
48     <version>${spring.version}</version>
49 </dependency>
50 <dependency>
51     <groupId>org.springframework</groupId>
52     <artifactId>spring-beans</artifactId>
53     <version>${spring.version}</version>
54 </dependency>
55 <dependency>
56     <groupId>org.springframework</groupId>
57     <artifactId>spring-context</artifactId>
58     <version>${spring.version}</version>
59 </dependency>
60
61 <!-- Spring testing annotations like @Contex
62 <dependency>
63     <groupId>org.springframework</groupId>
64     <artifactId>spring-test</artifactId>
65     <version>${spring.version}</version>
66 </dependency>
67
68 </dependencies>
69 </project>

```

**Step 4:** Service and DAO layer interfaces and implementations

- Unit Testing Q&A (2)
- BDD Testing (4)
- Data Access Uni
- JUnit Mockito Sp
- JUnit Mockito
- Spring Con
- Unit Testing
- Part 1: Unit te
- Part 2: Mockit
- Part 3: Mockit
- Part 4: Mockit
- Part 5: Mockit
- Testing Spring T.
- 5 Java unit tes
- JUnit with Hamc
- Spring Boot in ui
- Other Interview Q&A 1
- Free Java Interview

## 16 Technical Key Areas

open all | close all

- Best Practice (6)
- Coding (26)
- Concurrency (6)
- Design Concepts (7)
- Design Patterns (11)
- Exception Handling (3)
- Java Debugging (21)
- Judging Experience I
- Low Latency (7)
- Memory Managemen
- Performance (13)
- QoS (8)
- Scalability (4)
- SDLC (6)
- Security (13)
- Transaction Managen

## Service Layer

```

1 package com.mytutorial;
2
3 public interface SimpleService {
4     abstract String processUser(int id);
5 }

```

```

1 package com.mytutorial;
2
3 import javax.inject.Inject;
4 import javax.inject.Named;
5
6 @Named
7 public class SimpleServiceImpl implements SimpleService {
8
9     @Inject
10    SimpleDao dao;
11
12    public String processUser(int id) {
13        String name = dao.getNameById(id);
14        return "Hello " + name;
15    }
16 }

```

## DAO Layer

```

1 package com.mytutorial;
2
3 public interface SimpleDao {
4     abstract String getNameById(int id);
5 }

```

```

1 package com.mytutorial;
2
3 import javax.inject.Named;
4
5 @Named
6 public class SimpleDaoImpl implements SimpleDao {
7
8     public String getNameById(int id) {
9         if(id == 1) {
10            return "John";
11        }
12        else if (id == 2) {
13            return "Peter";
14        }
15        else {
16            throw new IllegalArgumentException();
17        }
18    }
19 }
20
21

```

## 80+ step by step Java Tutorials

[open all](#) | [close all](#)

- [Setting up Tutorial \(6\)](#)
- [Tutorial - Diagnosis \(2\)](#)
- [Akka Tutorial \(9\)](#)
- [Core Java Tutorials \(2\)](#)
- [Hadoop & Spark Tuto](#)
- [JEE Tutorials \(19\)](#)
- [Scala Tutorials \(1\)](#)
- [Spring & Hibernate Ti](#)
- [Tools Tutorials \(19\)](#)
- [Other Tutorials \(45\)](#)

## 100+ Java pre-interview coding tests

[open all](#) | [close all](#)

- [Can you write code? \(1\)](#)
- [Complete the given](#)
- [Converting from A to I](#)
- [Designing your classe](#)
- [Java Data Structures](#)
- [Passing the unit tests](#)
- [What is wrong with th](#)
- [Writing Code Home A](#)
- [Written Test Core Jav](#)
- [Written Test JEE \(1\)](#)

## How good are your .....?

[open all](#) | [close all](#)

- [Career Making Know](#)
- [Job Hunting & Resum](#)

### Step 5: Wiring the DI via Java config.

```
1 package com.mytutorial;
2
3 import org.springframework.context.annotation.Co
4 import org.springframework.context.annotation.Co
5
6 @Configuration
7 @ComponentScan(basePackages={"com.mytutorial"})
8 public class AppConfig {
9
10 }
```

**Step 6:** The JUnit test class. This is more of a **JUnit integration testing** than unit testing as the test spans service and dao layers. In Part 2, I will be mocking the Dao layer with the Mockito framework

```
1 package com.mytutorial;
2
3 import javax.inject.Inject;
4
5 import org.junit.Assert;
6 import org.junit.Test;
7 import org.junit.runner.RunWith;
8 import org.springframework.test.context.ContextC
9 import org.springframework.test.context.junit4.S
10
11 @ContextConfiguration(classes = { AppConfig.class
12 @RunWith(SpringJUnit4ClassRunner.class)
13 public class SimpleTest {
14
15     @Inject
16     SimpleService service;
17
18     @Test
19     public void testProcessUser(){
20         String processUser = service.processUser
21         Assert.assertEquals("Hello John", proces
22     }
23
24 }
25
```

**Step 7:** Run as JUnit test.

## Popular Member Posts

♦ 11 Spring boot interview questions & answers

906 views

♦ Q11-Q23: Top 50+ Core on Java OOP Interview Questions & Answers

816 views

001A: ♦ 7+ Java integration styles & patterns  
interview questions & answers

427 views

18 Java scenarios based interview Questions and  
Answers

409 views

♦ 7 Java debugging interview questions & answers

324 views

01b: ♦ 13 Spring basics Q8 – Q13 interview questions  
& answers

312 views

01: ♦ 15 Ice breaker questions asked 90% of the time  
in Java job interviews with hints

304 views

♦ 10 ERD (Entity-Relationship Diagrams) Interview  
Questions and Answers

301 views

♦ Q24-Q36: Top 50+ Core on Java classes, interfaces  
and generics interview questions & answers

251 views

♦ Object equals Vs == and pass by reference Vs  
value

234 views

0

Like

Share

Tweet

submit

reddit

0

G+1

1

Share

Bio

Latest Posts



## Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via [Amazon.com](http://Amazon.com) in 2005, and sold 35,000+ copies. Books are



outdated and replaced with this subscription based site. **945+** paid members. [join my LinkedIn Group](#). [Reviews](#)



#### About [Arulkumaran Kumaraswamipillai](#)

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via [Amazon.com](#) in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site. **945+** paid members. [join my LinkedIn Group](#). [Reviews](#)

◀ [CDI annotations @Named and @Inject Dependency Injection in Spring](#)

[3.0 Tutorial](#)

[Part 2: Mockito to fully mock the DAO layer](#) ▶

**Posted in** [JUnit Mockito Spring](#)

## Empowers you to open more doors, and fast-track

### Technical Know Hows

☀ [Java generics in no time](#) ☀ [Top 6 tips to transforming your thinking from OOP to FP](#) ☀ [How does a HashMap internally work? What is a hashing function?](#)  
☀ [10+ Java String class interview Q&As](#) ☀ [Java auto un/boxing benefits & caveats](#) ☀ [Top 11 slacknesses that can come back and bite you as an experienced Java developer or architect](#)

### Non-Technical Know Hows

☀ [6 Aspects that can motivate you to fast-track your career & go places](#) ☀ [Are you reinventing yourself as a Java developer?](#) ☀ [8 tips to safeguard your Java career against offshoring](#) ☀ [My top 5 career mistakes](#)

## Prepare to succeed

☀ [Turn readers of your Java CV go from “Blah blah” to “Wow”?](#) ☀ [How to prepare for Java job interviews?](#) ☀ [16 Technical Key Areas](#) ☀ [How to choose from multiple Java job offers?](#)

Select Category ▼

## © Disclaimer

The contents in this Java-Success are copy righted. The author has the right to correct or enhance the current content without any prior notice.

These are general advice only, and one needs to take his/her own circumstances into consideration. The author will not be held liable for any damages caused or alleged to be caused either directly or indirectly by these materials and resources. Any trademarked names or labels used in this blog remain the property of their respective trademark owners. No guarantees are made regarding the accuracy or usefulness of content, though I do make an effort to be accurate. Links to external sites do not imply endorsement of the linked-to sites.