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.

```
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.

```
<project xmlns="http://maven.apache.org/POM/4.0.
    xsi:schemaLocation="http://maven.apache.org/
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.mytutorial</groupId>
    <artifactId>simple-mocking-test</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>simple-mocking-test</name>
    <url>http://maven.apache.org</url>

    <properties>
        <project.build.sourceEncoding>UTF-8</pro
        <mockito-all.version>1.9.0</mockito-all.
        <spring.version>3.2.13.RELEASE</spring.v
    </properties>

    <dependencies>

    <!-- JUnit Library -->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.11</version>
            <scope>test</scope>
        </dependency>

        <!-- Mockito Library (for Mocking)-->
        <dependency>
            <groupId>org.mockito</groupId>
            <artifactId>mockito-all</artifactId>
            <version>${mockito-all.version}</ver
            <scope>test</scope>
        </dependency>

        <!-- Dependnecy Injection Annotation @Na
        <dependency>
            <groupId>javax.inject</groupId>
            <artifactId>javax.inject</artifactId
            <version>1</version>
        </dependency>

        <!-- Spring Dependencies (core libraries
        <dependency>
            <groupId>org.springframework</groupI
            <artifactId>spring-core</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupI
            <artifactId>spring-beans</artifactId
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupI
            <artifactId>spring-context</artifact
            <version>${spring.version}</version>
        </dependency>

    <!-- Spring testing annotations like @Contex
    <dependency>
            <groupId>org.springframework</groupI
            <artifactId>spring-test</artifactId>
            <version>${spring.version}</version>
        </dependency>

    </dependencies>
</project>
```

Step 4: Service and DAO layer interfaces and implementations

16 Technical Key Areas

Service Layer

```
package com.mytutorial;

public interface SimpleService {
    abstract String processUser(int id);
}
```

```
package com.mytutorial;

import javax.inject.Inject;
import javax.inject.Named;

@Named
public class SimpleServiceImpl implements Simple

    @Inject
    SimpleDao dao;

    public String processUser(int id) {
        String name = dao.getNameById(id);
        return "Hello " + name;
    }
}
```

DAO Layer

```
package com.mytutorial;

public interface SimpleDao {
    abstract String getNameById(int id);
}
```

```
package com.mytutorial;

import javax.inject.Named;

@Named
public class SimpleDaoImpl implements SimpleDao

    public String getNameById(int id) {
        if(id == 1) {
            return "John";
        }
        else if (id == 2) {
            return "Peter";
        }
        else {
            throw new IllegalArgumentException()
        }
    }
}


```

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 Tu
- Tools Tutorials (19)
- Other Tutorials (45)

100+ Java pre-interview coding tests

open all | close all

- Can you write code? (
- ♦ 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.

```
package com.mytutorial;

import org.springframework.context.annotation.Co
import org.springframework.context.annotation.Co

@Configuration
@ComponentScan(basePackages={"com.mytutorial"})
public class AppConfig {

}
```

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

```
package com.mytutorial;

import javax.inject.Inject;

import org.junit.Assert;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.test.context.ContextC
import org.springframework.test.context.junit4.S

@ContextConfiguration(classes = { AppConfig.clas
@RunWith(SpringJUnit4ClassRunner.class)
public class SimpleTest {

    @Inject
    SimpleService service;

    @Test
    public void testProcessUser(){
        String processUser = service.processUser
        Assert.assertEquals("Hello John", proces
    }

}

```

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 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