

IN 28 MINUTES



# Mockito for Beginner in 5 Steps

Mockito is the most popular mocking framework in Java.

## Git Repository

- <https://github.com/in28minutes/getting-started-in-5-steps/tree/master/mockito-in-5-steps>

## Pre-requisites

- Java & Eclipse – [https://www.youtube.com/playlist?list=PLBBog2r6uMCSmMVTW\\_QmDLyASBvovyAO3](https://www.youtube.com/playlist?list=PLBBog2r6uMCSmMVTW_QmDLyASBvovyAO3)
- JUnit – <https://courses.in28minutes.com/p/junit-tutorial-for-beginners>

## IDE Configuration

Easier Static Imports

- Window > Preferences > Java > Editor > Content Assist > Favorites
  - org.junit.Assert
  - org.mockito.BDDMockito
  - org.mockito.Mockito
  - org.hamcrest.Matchers
  - org.hamcrest.CoreMatchers

## Reference

- Visit Mockito Official Documentation – <http://site.mockito.org/mockito/docs/current/org/mockito/Mockito.html>

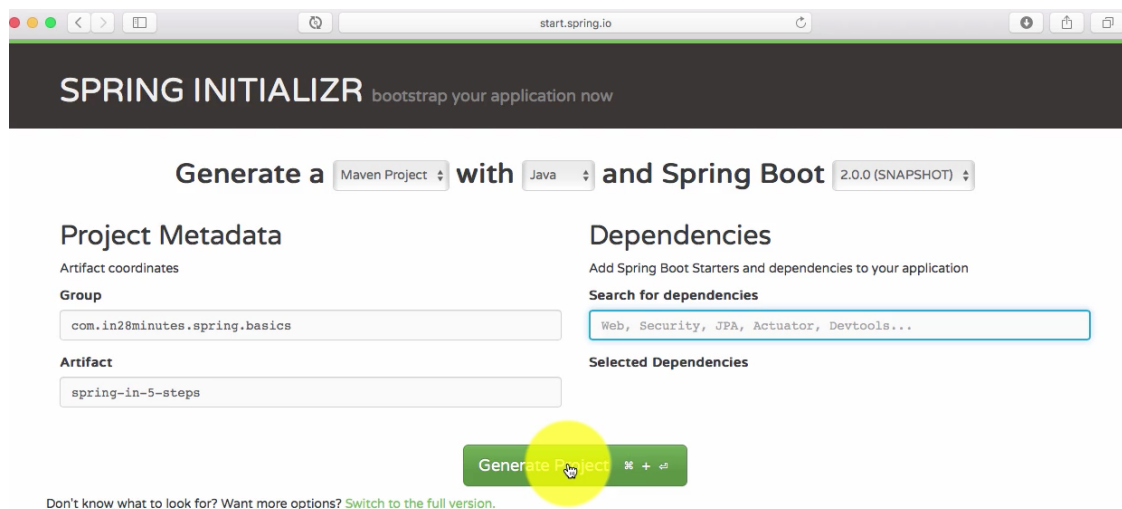
## Overview

- Step 1 : Setting up an example using <http://start.spring.io>.
- Step 2 : Using a Stubs - Disadvantages
- Step 3 : Your first mock.
- Step 4 : Using Mockito Annotations – @Mock, @InjectMocks, @RunWith(MockitoJUnitRunner.class)
- Step 5 : Mocking List interface

## Step 1 : Setting up an example using <http://start.spring.io>.

Creating a Spring Project with Spring Initializr is a cake walk.

Spring Initializr <http://start.spring.io/> is great tool to bootstrap your Spring Boot projects.



As shown in the image above, following steps have to be done

- Launch Spring Initializr and choose the following

[LIMITED TIME ONLY] Get upto 95% OFF on all Courses

Learn from the Best

- Choose Dependency
  - Web
- Click Generate Project.
- Import the project into Eclipse.
- If you want to understand all the files that are part of this project, you can go here.

Let's also set up a simple business class with a dependency.

- SomeBusinessImpl depends on DataService for the data
- findTheGreatestFromAllData has some business logic to find the greatest

```
package com.in28minutes.mockito.mockitodemo;

public class SomeBusinessImpl {
    private DataService dataService;

    public SomeBusinessImpl(DataService dataService) {
        super();
        this.dataService = dataService;
    }

    int findTheGreatestFromAllData() {
        int[] data = dataService.retrieveAllData();
        int greatest = Integer.MIN_VALUE;

        for (int value : data) {
            if (value > greatest) {
                greatest = value;
            }
        }
        return greatest;
    }
}

package com.in28minutes.mockito.mockitodemo;

public interface DataService {
    int[] retrieveAllData();
}
```

## Step 2 : Using a Stubs - Disadvantages

Let's use a stub to write a unit test for SomeBusinessImpl.

```
package com.in28minutes.mockito.mockitodemo;

import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class SomeBusinessStubTest {
    @Test
    public void testFindTheGreatestFromAllData() {
        SomeBusinessImpl businessImpl = new SomeBusinessImpl(new DataServiceStub());
        int result = businessImpl.findTheGreatestFromAllData();
        assertEquals(24, result);
    }
}

class DataServiceStub implements DataService {
    @Override
    public int[] retrieveAllData() {
        return new int[] { 24, 6, 15 };
    }
}
```

Problems with Stub

- How do I get DataServiceStub to return different data for different scenarios?
- Everytime DataService interface is updated with new methods, the DataServiceStub implementations should be updated.

## Step 3 : Your first mock.

```
package com.in28minutes.mockito.mockitodemo;

import static org.junit.Assert.assertEquals;
import static org.mockito.Mockito.mock;
import static org.mockito.Mockito.when;

import org.junit.Test;

public class SomeBusinessMockTest {

    @Test
    public void testFindTheGreatestFromAllData() {
        DataService dataServiceMock = mock(DataService.class);
        when(dataServiceMock.retrieveAllData()).thenReturn(new int[] { 24, 15, 3 });
        SomeBusinessImpl businessImpl = new SomeBusinessImpl(dataServiceMock);
        int result = businessImpl.findTheGreatestFromAllData();
        assertEquals(24, result);
    }
}
```

```

    }

    @Test
    public void testFindTheGreatestFromAllData_ForOneValue() {
        DataService dataServiceMock = mock(DataService.class);
        when(dataServiceMock.retrieveAllData()).thenReturn(new int[] { 15 });
        SomeBusinessImpl businessImpl = new SomeBusinessImpl(dataServiceMock);
        int result = businessImpl.findTheGreatestFromAllData();
        assertEquals(15, result);
    }
}

```

## Notes

- `DataService dataServiceMock = mock(DataService.class)` - We are using the mock method to create a mock.
- `when(dataServiceMock.retrieveAllData()).thenReturn(new int[] { 24, 15, 3 })` - stubbing the mock to return specific data

## Step 4 : Using Mockito Annotations - @Mock, @InjectMocks, @RunWith(MockitoJUnitRunner.class)

```

package com.in28minutes.mockito.mockitodemo;

import static org.junit.Assert.assertEquals;
import static org.mockito.Mockito.when;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.junit.MockitoJUnitRunner;

@RunWith(MockitoJUnitRunner.class)
public class SomeBusinessMockAnnotationsTest {

    @Mock
    DataService dataServiceMock;

    @InjectMocks
    SomeBusinessImpl businessImpl;

    @Test
    public void testFindTheGreatestFromAllData() {
        when(dataServiceMock.retrieveAllData()).thenReturn(new int[] { 24, 15, 3 });
        assertEquals(24, businessImpl.findTheGreatestFromAllData());
    }

    @Test
    public void testFindTheGreatestFromAllData_ForOneValue() {
        when(dataServiceMock.retrieveAllData()).thenReturn(new int[] { 15 });
        assertEquals(15, businessImpl.findTheGreatestFromAllData());
    }

    @Test
    public void testFindTheGreatestFromAllData_NoValues() {
        when(dataServiceMock.retrieveAllData()).thenReturn(new int[] {});
        assertEquals(Integer.MIN_VALUE, businessImpl.findTheGreatestFromAllData());
    }
}

```

## Notes

- `@Mock DataService dataServiceMock;` - Create a mock for DataService.
- `@InjectMocks SomeBusinessImpl businessImpl;` - Inject the mocks as dependencies into businessImpl.
- `@RunWith(MockitoJUnitRunner.class)` - The JUnit Runner which causes all the initialization magic with `@Mock` and `@InjectMocks` to happen before the tests are run.

## Step 5 : Mocking List interface

Mocking a method. Mock returns the same value on multiple calls.

```

@Test
public void testSize() {
    List listMock = mock(List.class);
    when(listMock.size()).thenReturn(10);
    assertEquals(10, listMock.size());
    assertEquals(10, listMock.size());
}

```

Setting the mock to return 10 on first call and 20 on the second call.

```

@Test
public void testSize_multipleReturns() {
    List listMock = mock(List.class);
    when(listMock.size()).thenReturn(10).thenReturn(20);
    assertEquals(10, listMock.size());
    assertEquals(20, listMock.size());
    assertEquals(20, listMock.size());
}

```

Customizing the mock based on a specific parameter value.

```
@Test
public void testGet_SpecificParameter() {
    List listMock = mock(List.class);
    when(listMock.get(0)).thenReturn("SomeString");
    assertEquals("SomeString", listMock.get(0));
    assertEquals(null, listMock.get(1));
}
```

Using a generic argument – Mockito.anyInt()

```
@Test
public void testGet_GenericParameter() {
    List listMock = mock(List.class);
    when(listMock.get(Mockito.anyInt())).thenReturn("SomeString");
    assertEquals("SomeString", listMock.get(0));
    assertEquals("SomeString", listMock.get(1));
}
```

## Complete Code Example

### /pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.in28minutes.mockito</groupId>
    <artifactId>mockito-demo</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>mockito-demo</name>
    <description>Demo project for Spring Boot</description>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.0.0.RELEASE</version>
        <relativePath/> <!-- Lookup parent from repository -->
    </parent>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
        <java.version>1.8</java.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

    <repositories>
        <repository>
            <id>spring-snapshots</id>
            <name>Spring Snapshots</name>
            <url>https://repo.spring.io/snapshot</url>
            <snapshots>
                <enabled>true</enabled>
            </snapshots>
        </repository>
        <repository>
            <id>spring-milestones</id>
            <name>Spring Milestones</name>
            <url>https://repo.spring.io/milestone</url>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </repository>
    </repositories>

    <pluginRepositories>
        <pluginRepository>
            <id>spring-snapshots</id>
            <name>Spring Snapshots</name>
            <url>https://repo.spring.io/snapshot</url>
```

```

        <snapshots>
            <enabled>true</enabled>
        </snapshots>
    </pluginRepository>
    <pluginRepository>
        <id>spring-milestones</id>
        <name>Spring Milestones</name>
        <url>https://repo.spring.io/milestone</url>
        <snapshots>
            <enabled>>false</enabled>
        </snapshots>
    </pluginRepository>
</pluginRepositories>

</project>

```

---

## /src/main/java/com/in28minutes/mockito/mockitodemo/DataService.java

```

package com.in28minutes.mockito.mockitodemo;

public interface DataService {
    int[] retrieveAllData();
}

```

---

## /src/main/java/com/in28minutes/mockito/mockitodemo/MockitoDemoApplication.java

```

package com.in28minutes.mockito.mockitodemo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class MockitoDemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(MockitoDemoApplication.class, args);
    }

}

```

---

## /src/main/java/com/in28minutes/mockito/mockitodemo/SomeBusinessImpl.java

```

package com.in28minutes.mockito.mockitodemo;

public class SomeBusinessImpl {
    private DataService dataService;

    public SomeBusinessImpl(DataService dataService) {
        super();
        this.dataService = dataService;
    }

    int findTheGreatestFromAllData() {
        int[] data = dataService.retrieveAllData();
        int greatest = Integer.MIN_VALUE;

        for (int value : data) {
            if (value > greatest) {
                greatest = value;
            }
        }
        return greatest;
    }

}

```

---

## /src/main/resources/application.properties

---

## /src/test/java/com/in28minutes/mockito/mockitodemo/ListTest.java

```

package com.in28minutes.mockito.mockitodemo;

import static org.junit.Assert.assertEquals;
import static org.mockito.Mockito.mock;
import static org.mockito.Mockito.when;

```

```

import java.util.List;

import org.junit.Test;
import org.mockito.Mockito;

public class ListTest {

    @Test
    public void testSize() {
        List listMock = mock(List.class);
        when(listMock.size()).thenReturn(10);
        assertEquals(10, listMock.size());
        assertEquals(10, listMock.size());
    }

    @Test
    public void testSize_multipleReturns() {
        List listMock = mock(List.class);
        when(listMock.size()).thenReturn(10).thenReturn(20);
        assertEquals(10, listMock.size());
        assertEquals(20, listMock.size());
        assertEquals(20, listMock.size());
    }

    @Test
    public void testGet_SpecificParameter() {
        List listMock = mock(List.class);
        when(listMock.get(0)).thenReturn("SomeString");
        assertEquals("SomeString", listMock.get(0));
        assertEquals(null, listMock.get(1));
    }

    @Test
    public void testGet_GenericParameter() {
        List listMock = mock(List.class);
        when(listMock.get(Mockito.anyInt())).thenReturn("SomeString");
        assertEquals("SomeString", listMock.get(0));
        assertEquals("SomeString", listMock.get(1));
    }
}

```

---

## /src/test/java/com/in28minutes/mockito/mockitodemo/MockitoDemoApplicationTests

```

package com.in28minutes.mockito.mockitodemo;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit4.SpringRunner;

@RunWith(SpringRunner.class)
@SpringBootTest
public class MockitoDemoApplicationTests {

    @Test
    public void contextLoads() {
    }

}

```

---

## /src/test/java/com/in28minutes/mockito/mockitodemo/SomeBusinessMockAnnotation

```

package com.in28minutes.mockito.mockitodemo;

import static org.junit.Assert.assertEquals;
import static org.mockito.Mockito.when;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.junit.MockitoJUnitRunner;

@RunWith(MockitoJUnitRunner.class)
public class SomeBusinessMockAnnotationsTest {

    @Mock
    DataService dataServiceMock;

    @InjectMocks
    SomeBusinessImpl businessImpl;

    @Test
    public void testFindTheGreatestFromAllData() {
        when(dataServiceMock.retrieveAllData()).thenReturn(new int[] { 24, 15, 3 });
        assertEquals(24, businessImpl.findTheGreatestFromAllData());
    }

    @Test
    public void testFindTheGreatestFromAllData_ForOneValue() {
        when(dataServiceMock.retrieveAllData()).thenReturn(new int[] { 15 });
        assertEquals(15, businessImpl.findTheGreatestFromAllData());
    }
}

```

```

@Test
public void testFindTheGreatestFromAllData_NoValues() {
    when(dataServiceMock.retrieveAllData()).thenReturn(new int[] {});
    assertEquals(Integer.MIN_VALUE, businessImpl.findTheGreatestFromAllData());
}
}

```

## /src/test/java/com/in28minutes/mockito/mockitodemo/SomeBusinessMockTest.java

```

package com.in28minutes.mockito.mockitodemo;

import static org.junit.Assert.assertEquals;
import static org.mockito.Mockito.mock;
import static org.mockito.Mockito.when;

import org.junit.Test;

public class SomeBusinessMockTest {

    @Test
    public void testFindTheGreatestFromAllData() {
        DataService dataServiceMock = mock(DataService.class);
        when(dataServiceMock.retrieveAllData()).thenReturn(new int[] { 24, 15, 3 });
        SomeBusinessImpl businessImpl = new SomeBusinessImpl(dataServiceMock);
        int result = businessImpl.findTheGreatestFromAllData();
        assertEquals(24, result);
    }

    @Test
    public void testFindTheGreatestFromAllData_ForOneValue() {
        DataService dataServiceMock = mock(DataService.class);
        when(dataServiceMock.retrieveAllData()).thenReturn(new int[] { 15 });
        SomeBusinessImpl businessImpl = new SomeBusinessImpl(dataServiceMock);
        int result = businessImpl.findTheGreatestFromAllData();
        assertEquals(15, result);
    }
}

```

## /src/test/java/com/in28minutes/mockito/mockitodemo/SomeBusinessStubTest.java

```

package com.in28minutes.mockito.mockitodemo;

import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class SomeBusinessStubTest {

    @Test
    public void testFindTheGreatestFromAllData() {
        SomeBusinessImpl businessImpl = new SomeBusinessImpl(new DataServiceStub());
        int result = businessImpl.findTheGreatestFromAllData();
        assertEquals(24, result);
    }

}

class DataServiceStub implements DataService {
    @Override
    public int[] retrieveAllData() {
        return new int[] { 24, 6, 15 };
    }
}

```

Join our free Spring Boot in 10 Steps Course.

Find out how in28Minutes reached 100,000 Learners on Udemy in 2 years. [The in28minutes Way](#) – Our approach to creating awesome learning experiences.

