

```

1  =====pom.xml=====
2  <?xml version="1.0" encoding="UTF-8"?>
3  <project xmlns="http://maven.apache.org/POM/4.0.0"
4  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
6  https://maven.apache.org/xsd/maven-4.0.0.xsd">
7  <modelVersion>4.0.0</modelVersion>
8  <parent>
9  <groupId>org.springframework.boot</groupId>
10 <artifactId>spring-boot-starter-parent</artifactId>
11 <version>2.3.0.RELEASE</version>
12 <relativePath/> <!-- lookup parent from repository -->
13 </parent>
14 <groupId>com.ameya</groupId>
15 <artifactId>009-SpringBootJUnitMockito</artifactId>
16 <version>0.0.1-SNAPSHOT</version>
17 <name>009-SpringBootJUnitMockito</name>
18 <description>Demo project for Spring Boot</description>
19 <properties>
20 <java.version>1.8</java.version>
21 </properties>
22 <dependencies>
23 <dependency>
24 <groupId>org.springframework.boot</groupId>
25 <artifactId>spring-boot-starter-web</artifactId>
26 </dependency>
27
28 <dependency>
29 <groupId>org.springframework.boot</groupId>
30 <artifactId>spring-boot-starter-test</artifactId>
31 <scope>test</scope>
32 <!--
33 <exclusions>
34 <exclusion>
35 <groupId>org.junit.vintage</groupId>
36 <artifactId>junit-vintage-engine</artifactId>
37 </exclusion>
38 </exclusions>
39 <!-->
40 </dependency>
41 </dependencies>
42
43 <build>
44 <plugins>
45 <plugin>
46 <groupId>org.springframework.boot</groupId>
47 <artifactId>spring-boot-maven-plugin</artifactId>
48 </plugin>
49 </plugins>
50 </build>
51 </project>
52 =====MyDAO.java=====
53 package com.ameya.daos;
54
55 import org.springframework.stereotype.Repository;
56
57 @Repository
58 public class MyDAO {
59
60     public int[] retrieveAllData() {
61         return new int[] {1,2,3,4,5};
62     }
63 }
64 =====MyService.java=====
65 package com.ameya.services;
66
67 import org.springframework.beans.factory.annotation.Autowired;
68 import org.springframework.stereotype.Service;
69
70 import com.ameya.daos.MyDAO;
71
72 @Service

```

```

72 public class MyService {
73     @Autowired
74     private MyDAO dataService;
75
76     public int findTheGreatest() {
77         int[] data=dataService.retrieveAllData();
78         int greatest=Integer.MIN_VALUE;
79         for(int value : data) {
80             if(value > greatest) {
81                 greatest=value;
82             }
83         }
84         return greatest;
85     }
86 }
87 =====MyServiceMockSpringContextTest.java=====
88 package com.ameya.unittests;
89
90 import static org.junit.Assert.assertEquals;
91 import static org.mockito.Mockito.when;
92
93 import org.junit.Test;
94 import org.junit.runner.RunWith;
95 import org.springframework.beans.factory.annotation.Autowired;
96 import org.springframework.boot.test.context.SpringBootTest;
97 import org.springframework.boot.test.mock.mockito.MockBean;
98 import org.springframework.test.context.junit4.SpringRunner;
99
100 import com.ameya.daos.MyDAO;
101 import com.ameya.services.MyService;
102
103 @RunWith(SpringRunner.class)
104 @SpringBootTest
105 public class MyServiceMockSpringContextTest {
106
107     @MockBean
108     MyDAO dataServiceMock;
109     @Autowired
110     MyService businessImpl;
111
112     @Test
113     public void testFindTheGreatestFromAllData() {
114         when(dataServiceMock.retrieveAllData()).thenReturn(new int[] {24,15,3});
115         assertEquals(24,businessImpl.findTheGreatest());
116     }
117     @Test
118     public void testFindTheGreatestFromAllDataForOneValue() {
119         when(dataServiceMock.retrieveAllData()).thenReturn(new int[] {15});
120         assertEquals(15,businessImpl.findTheGreatest());
121     }
122     @Test
123     public void testFindTheGreatestFromAllDataForNoValues() {
124         when(dataServiceMock.retrieveAllData()).thenReturn(new int[] {});
125         assertEquals(Integer.MIN_VALUE,businessImpl.findTheGreatest());
126     }
127 }
128 =====MyServiceMockTest.java=====
129 package com.ameya.unittests;
130
131 import static org.junit.Assert.assertEquals;
132 import static org.mockito.Mockito.when;
133
134 import org.junit.Test;
135 import org.junit.runner.RunWith;
136 import org.mockito.InjectMocks;
137 import org.mockito.Mock;
138 import org.mockito.junit.MockitoJUnitRunner;
139
140 import com.ameya.daos.MyDAO;
141 import com.ameya.services.MyService;
142
143 @RunWith(MockitoJUnitRunner.class)
144 public class MyServiceMockTest {

```

```
145
146     @Mock
147     MyDAO dataServiceMock;
148     @InjectMocks
149     MyService businessImpl;
150     @Test
151     public void testFindTheGreatestFromAllData() {
152         when(dataServiceMock.retrieveAllData()).thenReturn(new int[] {24,15,3});
153         assertEquals(24,businessImpl.findTheGreatest());
154     }
155     @Test
156     public void testFindTheGreatestFromAllDataForOneValue() {
157         when(dataServiceMock.retrieveAllData()).thenReturn(new int[] {15});
158         assertEquals(15,businessImpl.findTheGreatest());
159     }
160     @Test
161     public void testFindTheGreatestFromAllDataForNoValues() {
162         when(dataServiceMock.retrieveAllData()).thenReturn(new int[] {});
163         assertEquals(Integer.MIN_VALUE,businessImpl.findTheGreatest());
164     }
165 }
166
```