Date : 25/03/2021
Spring Boot 9AM
Mr. RAGHU
---

Workspace:
https://www.mediafire.com/file/erebpiaythelraj/SpringBoot9AM_25032021_WORKSPACE_RAGHU.zip/file

## Unit Testing in Spring Boot ReST

JUnit#1:
https://www.youtube.com/watch?v=PT9WQ_Rz1ew
JUnit#2:
https://www.youtube.com/watch?v=Rue28g3reRI
Eclipse Debug#
https://www.youtube.com/watch?v=HwwF4pvYWws

==========

Project => Modules , Module => RestController + SL + DAL + Model

*) Test Module
  a. POSTMAN Tool
  b. Swagger UI **
  c. JUnit with Mock **

=> Module written by developer, must be tested.
=> Testing by using Programming is called as JUnit 5.x
=> By using JUnit we should define TestCase(Program/class)
   used to test our code.

=> JUnit 5 Annotations
  @Test
  @BeforeEach
  @AfterEach
  @Disabled
  @Order
  ..etc

*) Mock/Mocking : Creating Proxy Stubs for test supported modules.
   Mocking is a process of creating dummy impl class + object
         + inject to actual services/modules.

=> For our Module Testing what are all dependent modules exist
   they are stubbed (Proxy) by Mocking and injected to out Module.

=> *** Why Mocking used with JUnit? Partial Testing/Module Testing

```
<dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-engine</artifactId>
    <version>5.6.2</version>
    <scope>test</scope>
</dependency>

<dependency>
    <groupId>org.mockito</groupId>
```

```
    <artifactId>mockito-core</artifactId>
    <version>3.3.3</version>
    <scope>test</scope>
</dependency>

<dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-junit-jupiter</artifactId>
    <version>3.3.3</version>
    <scope>test</scope>
</dependency>

*) Mockito(C) has static method mock(T.class):T object
   it will stub your given class and creates proxy object.

Ex#
interface A{
  int show();
}
A  oa = mock(A.class);

Internal work of mock() method:
S#1 Creating Impl class with default returns of methods
class $1 implements A{
  public int show() {
    return 0;
  }
}
S#2 Creating object and cast to same type
A oa = new $1();

----------------------------
Ex#2
package in.nit;
public interface CalculateService {

        double getFinalCost(
                        double basecost,
                        int qty,
                        double disc);

}

Mocking sample code:
CalculateService cs = mock(CalculateService.class);
when(cs.getFinalCost(basecost, qty, disc))
   .thenReturn((basecost-disc)*qty);

Generated class
class $1 implements CalculateService {
  public double getFinalCost(
        double basecost, int qty, double disc)
  {
     return (basecost-disc)*qty;
  }
}
CalculateService cs = new $1();
```

```
----------------------------Full code-----------
package in.nit;

public interface CalculateService {

	double getFinalCost(
			double basecost,
			int qty,
			double disc);

}
------------
package in.nit;

public class ProductService {

	private CalculateService cs;

	public void setCs(CalculateService cs) {
		this.cs = cs;
	}

	public String getFinalMsg(
			double basecost,
			int qty,
			double disc)
	{
		return "Total cost is " +
				cs.getFinalCost(basecost,qty,disc);
	}

}



===Test case#1 without anntations===
package in.nit;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.mockito.Mockito.mock;
import static org.mockito.Mockito.when;

import org.junit.jupiter.api.Test;

public class TestCalcukateService {

	@Test
	public void testGetFinalCost() {
		String expected ="Total cost is 1000.0";
		double basecost = 200.0;
		int qty=10;
		double disc = 100.0;

		CalculateService cs = mock(CalculateService.class);
		when(cs.getFinalCost(basecost, qty, disc))
		.thenReturn((basecost-disc)*qty);
```

```
                ProductService ps = new ProductService();
                ps.setCs(cs);
                String actual = ps.getFinalMsg(basecost, qty, disc);
                assertEquals(expected, actual);
        }
}
======Testcase with annotations=======
@Mock
private CalculateService cs;
  => create impl class  + object

@InjectMocks
private ProductService ps; => create ps object with cs object

-------------------------------------------------
package in.nit;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.mockito.Mockito.when;

import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.MockitoAnnotations;

public class TestCalcukateServiceTwo {

        @Mock
        private CalculateService cs;
        @InjectMocks
        private ProductService ps;

        private String expected ;
        private double basecost;
        private int qty;
        private double disc ;
        private String actual;

        @BeforeEach
        public void setup() {
                //activate Mocking Annotations
                MockitoAnnotations.initMocks(this);
                expected ="Total cost is 1000.0";
                basecost = 200.0;
                qty = 10;
                disc = 100.0;
                //ps = new ProductService();
        }

        @Test
        public void testGetFinalCost() {
                when(cs.getFinalCost(basecost, qty, disc))
                .thenReturn((basecost-disc)*qty);

                //ps.setCs(cs);
```

```
                actual = ps.getFinalMsg(basecost, qty, disc);
                assertEquals(expected, actual);
        }

        @AfterEach
        public void clean() {
                cs = null;
                basecost = disc = 0.0;
                qty = 0;
                expected = actual = null;

        }
}
```