```
                         Date : 27/03/2021
                         Spring Boot 9AM
                           Mr. RAGHU
                 ------------------------------
Eclipse Debug:
https://www.youtube.com/watch?v=HwwF4pvYWws

Spring AOP:
https://www.youtube.com/c/NareshIT/search?query=Spring%20AOP

                   Spring Boot Unit Testing
                     (JUnit with Mockito)

=> /src/test/java
    |-Demo.java

---one time setup-------------
1. Loading Properties file for Test environment
  @TestPropertySource("classpath:application-test.properties")

2. Spring Container for Web Env

 @SpringBootTest(webEnvironment = WebEnvironment.MOCK)

3. Create all required objects in container by using AutoConfiguration

 @AutoConfigureMockMvc
_________________________________________________________
 To Test one operation

 a. Create one Dummy/proxy(not made by client) Http Request
 b. Execute dummy request using MockMvc and get result(MvcResult)
 c. Read Response from result object
 d. Assert Response Data(checking Status, Body/content, header
types...etc)


--Try for PUT Update Test cases--
Task#1 (Refer save test case)
create Request
    PUT("/employee/update")
    contentType("app/json")
    content({empId:10...})

Assert status and response(updated)-if not fail

--Try for PATCH Update Test cases--
Task#2 (refer get test case)
create Request
    patch("/employee/modify/10/abcd@gmail.com")

Assert status and response(updated)-if not fail
===============Full code=====================
package in.nareshit.raghu;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertNotNull;
import static org.junit.jupiter.api.Assertions.fail;

import org.junit.jupiter.api.Disabled;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Order;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureM
ockMvc;
import org.springframework.boot.test.context.SpringBootTest;
import
org.springframework.boot.test.context.SpringBootTest.WebEnvironment;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.mock.web.MockHttpServletResponse;
import org.springframework.test.context.TestPropertySource;
import org.springframework.test.web.servlet.MockMvc;
import org.springframework.test.web.servlet.MvcResult;
import
org.springframework.test.web.servlet.request.MockHttpServletRequestBui
lder;
import
org.springframework.test.web.servlet.request.MockMvcRequestBuilders;

@SpringBootTest(webEnvironment = WebEnvironment.MOCK)
@TestPropertySource("classpath:application-test.properties")
@AutoConfigureMockMvc
public class SpringBoot2RestCrudMySqlApplicationTests {

        @Autowired
        private MockMvc mockMvc;

        @Test
        @DisplayName("EMPLOYEE SAVE")
        @Order(1)

        public void testSaveEmployee() throws Exception {
                //1. request object
                MockHttpServletRequestBuilder request =
                                MockMvcRequestBuilders
                                .post("/employee/save")
                                .contentType("application/json")
                                .content("
{\"empName\":\"ABCDEFGH\",\"empMail\":\"abcd@gmail.com\",\"empSal\":
20000.0}");

                //2. execute and get result
                MvcResult result =
mockMvc.perform(request).andReturn();

                //3. read response
                MockHttpServletResponse response =
result.getResponse();

                //4. validate(assert) response details
```

```
		assertEquals(HttpStatus.CREATED.value(), 
response.getStatus());
		if(!response.getContentAsString().contains("Employee 
saved")) {
			fail("Employee not created");
		}
	}

	@Test
	@DisplayName("FETCH ONE EMPLOYEE")
	@Order(2)

	public void testGetOneEmployee() throws Exception {
		//1. request creation
		MockHttpServletRequestBuilder request = 

MockMvcRequestBuilders.get("/employee/find/10");

		//2. execute and get result
		MvcResult result = 
mockMvc.perform(request).andReturn();

		//3. read response
		MockHttpServletResponse response = 
result.getResponse();

		//4. validate data
		assertEquals(HttpStatus.OK.value(), 
response.getStatus());
		assertNotNull(response.getContentAsString());
		assertEquals(MediaType.APPLICATION_JSON_VALUE, 
response.getContentType());

	}


	@Test
	@DisplayName("FETCH ONE EMPLOYEE NOT FOUND")
	@Order(3)

	public void testGetOneEmployeeNotExist() throws Exception {
		//1. request creation
		MockHttpServletRequestBuilder request = 

MockMvcRequestBuilders.get("/employee/find/36");

		//2. execute and get result
		MvcResult result = 
mockMvc.perform(request).andReturn();

		//3. read response
		MockHttpServletResponse response = 
result.getResponse();

		//4. validate data
		assertEquals(HttpStatus.NOT_FOUND.value(), 
response.getStatus());
```

```
		assertNotNull(response.getContentAsString());
		if(!response.getContentAsString().contains("Not 
exist")) {
			fail("Employee Exist! Check it once!!");
		}
		
	}
	
	
	@Test
	@DisplayName("FETCH ALL EMPLOYEES")
	@Order(4)
	
	public void testAllEmployees() throws Exception {
		//1. request creation
		MockHttpServletRequestBuilder request = 

MockMvcRequestBuilders.get("/employee/all");
		
		//2. execute and get result
		MvcResult result = 
mockMvc.perform(request).andReturn();
		
		//3. read response
		MockHttpServletResponse response = 
result.getResponse();
		
		//4. validate data
		assertEquals(HttpStatus.OK.value(), 
response.getStatus());
		assertNotNull(response.getContentAsString());
		assertEquals(MediaType.APPLICATION_JSON_VALUE, 
response.getContentType());
		System.out.println(response.getContentAsString());
	}
	
	@Test
	@DisplayName("REMOVE ONE EMPLOYEE BY ID")
	@Order(5)
	public void testRemoveOneEmployee() throws Exception {
		
		MockHttpServletRequestBuilder request = 
				MockMvcRequestBuilders
				.delete("/employee/remove/10");
		
		//2. execute and get result
		MvcResult result = 
mockMvc.perform(request).andReturn();
		
		//3. read response
		MockHttpServletResponse response = 
result.getResponse();
		
		//4. validate
		assertEquals(HttpStatus.OK.value(), 
response.getStatus());
		if(!response.getContentAsString().contains("Employee
deleted")) {
                        fail("Unable to Delete");
                }
        }


}
```