Date : 26/03/2021
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
Redis cache:
https://www.youtube.com/watch?v=HBmlNMGh9O0
https://www.youtube.com/watch?v=IwYEdZOmY6g
Docker:
https://www.youtube.com/watch?v=LmoLFcoaeQw
https://www.youtube.com/watch?v=6_6MoohzdEI
PCF:
https://www.youtube.com/watch?v=QOwgiJWmZ9k

Spring Boot Unit Testing
(JUnit with Mockito)

Latest/ junit-jupiter engine : JUnit 5.x
Legacy/vintage versions are : JUnit 4.x + JUnit 3.x

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

spring-boot-starter-test = junit-jupiter + vintage + mockito

*) Spring Boot Latest Versions 2.3.x is recomanding us to use latest version of JUnit (ie JUnit 5.x). They are in plan to remove vintage engine in upcoming versions.
So, in pom.xml also they have provided exclusions for vintage engine.

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
        <exclusions>
                <exclusion>
                        <groupId>org.junit.vintage</groupId>
                        <artifactId>junit-vintage-engine</artifactId>
                </exclusion>
        </exclusions>
</dependency>
```

==========

Mocking: Creating Dummy impl/objects

Spring Boot Application?
-> Component Scanning
-> Auto Configuration for WebApp
-> Spring Container
  [AnnotationConfigServletWebApplicationContext]
-> Request objects
-> Environment details(props)

*) We need to provide all above details using Mock support.
  If run starter class that takes care of all above thing.
  But this time we are running Test class.

--Setup Full Environment (equal work to starter class)--------
1. Loading Properties file for Test environment
 *) If we do not provide this, by default : application.properties
   file is loaded. Else we can use :

   @TestPropertySource("classpath:application-test.properties")

2. Create Spring Container that uses internally WebApplication
   Support(DispatcherServlet)

   @SpringBootTest(webEnvironment = WebEnvironment.MOCK)

 *) It creates by default :
AnnotationConfigServletWebApplicationContext

3. Create objects required inside container
  a. DataSource
  b. HandlerMapping
  c. ORM Config (EntityManagerFactory, EntityManager)
   ..etc

   It can be implemented using : @AutoConfigureMockMvc

*) For the final runtime environment reference is given by MockMvc.
  That supports Test Dispatcher Servlet, Access Spring Container,
  execution of RestControllers, Database Connection support..etc
----
*) TODO Unit Test coding follow below steps
 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)

{"empName":"A","empSal":2500.0,"empMail":"a@gm.com"}

*) Database Dumps are used for Mock Testing,
   not the production or UAT env used.

------------Sample code--------------------
package in.nareshit.raghu;

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

	@Autowired
	private MockMvc mockMvc;

	@Test
	public void testSaveOp() throws Exception {
		//1. prepare http request
		MockHttpServletRequestBuilder request =
				MockMvcRequestBuilders
				.post("/employee/update")
				.contentType("application/json")
				.content("{empName:_..}")
				;

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

		//3. Read Response from result
		MockHttpServletResponse response =
result.getResponse();

		//4. assert/validate result
		assertEquals(HttpStatus.CREATED.value(),
response.getStatus());
		assertNotNull(response.getContentAsString());
	}

}
```