**Mockito - Mocking framework for unit tests in Java**

* Mocking for unit testing is when you create an object that implements the behavior of a real subsystem in controlled ways. In short, mocks are used as a replacement for a dependency.
* With Mockito, you create a mock, tell Mockito what to do when specific methods are called on it, and then use the mock instance in your test instead of the real thing. After the test, you can query the mock to see what specific methods were called or check the side effects in the form of changed state.
* By default, Mockito provides an implementation for every method of the mock.

**Spring Boot Application (CRUD Operations – Using h2 database) Workflow-**

**Controller**

**Service**

**Repository**

Database

**Spring Boot Application (CRUD Operations – Using Mockito) Workflow-**

**Mockito**

**Controller**

**Service**

**Repository**

**Database**

**Below are the method details we have created and mocked in our project-**

|  |
| --- |
| 1. **getAllEmployees()**   To get all employees saved in database  **Method-** Get  **API Endpoint-** http://localhost:8080/employees |
| 1. **getEmployeeById(Long id)**   To get employees by providing ID  **Method-** Get  **API Endpoint-** http://localhost:8080/employees/1 |
| 1. **createOrUpdateEmployee( EmployeeEntity entity)**   To create or Update employee based on employee id  **Method-** Post  **API** **Endpoint-** <http://localhost:8080/employees>  Request Body-  {  "id":"3",  "firstName":"Akash",  "lastName":"Sonawane",  "email":"akash.sonawane@gmail.com"  } |
| 1. **deleteEmployeeById(Long id)**   To delete employee by providing employee ID  **Method-** Delete  **API Endpoint-** <http://localhost:8080/employees/3> |

Below are the details of annotation and methods used in test class-

* @MockBean: (org.springframework.boot.test.mock.mockito.MockBean)

Annotation that can be used to add mocks to a Spring ApplicationContext. Can be used as test classes that are @RunWith the SpringRunner.

Mocks can be registered by type or by bean name. Any existing single bean of the same type defined in the context will be replaced by the mock, if no existing bean is defined a new one will be added.

When @MockBean is used on a field, as well as being registered in the application context, the mock will also be injected into the field.

* Mockito offers two ways of stubbing.
* The first way is “when this method is called, then do something.” Consider the following snippet:

when(passwordEncoder.encode("1")).thenReturn("a");

It reads almost like English: “When passwordEncoder.encode(“1”) is called, return an a.”

* The second way of stubbing reads more like “Do something when this mock’s method is called with the following arguments.” This way of stubbing is harder to read as the cause is specified at the end. Consider:

doReturn("a").when(passwordEncoder).encode("1");

The snippet with this method of stubbing would read: “Return a when passwordEncoder’s encode() method is called with an argument of 1.”

* [*verify()*](http://javadoc.io/doc/org.mockito/mockito-core/latest/org/mockito/Mockito.html#verify-T-)*:* to check methods were called with given arguments