**Question :** If it is possible to mock DB in JUnit test? Give a case to show how to do it.

Yes , We can mock the database part during Junit the code.

I have done in my project using Mockito .

Here is an use case :

1. Connect to the oracle database
2. Insert into table “endpointdetails” data like
   1. Service\_name
   2. Endpoint\_url
   3. Pagination\_value
   4. Offset\_value
   5. .
   6. .
   7. .
3. retrieve the data from “endpointdetails” table and pass the value to java controller

Java classes are EndpointController , EndpointService, EndpointServiceImpl, EndpointDao, EndpointBean.

We will do write the Junit for EndpointDao , which handles all the DB activities.

**Problem** : The **database is not ready**. But we still need to complete the unit test cases

Solution : We will use Mocito to mock the database connectivity and Mock some sample data to do the unit testing so that the DB functionalities will work as expected when ready.

Sample Mocks :

Here are some sample code that mocks the functionalities

@InjectMocks EndpointDao endpointDao;

@Mock Connection conn;

@Mock Statement stmt;

Test cases :

// This will return a mock statement

Mockito.when(conn.createStatement()).thenReturn(stmt);

//we can check the connection object is created once

Mockito.verify(conn.createStatement(), Mockito.times(1));

verify(conn, times(1)).prepareStatement(anyString(), anyInt());

Conclusion :

So the basic steps are

1. Crate a test class , and inject the dependency classes to test
2. Create Mock objects using @Mock annotations
3. Create mock data and add to mock objects
4. Create test method and using different versions of when() and verify() methods of mocito , call the service or execution path and check the results
5. We can also use the assert methods of Junit along with Mockito
6. PowerMockito is also another library to create mock objects and can be used along with mockito