

## **Spring Boot Microservices API Test using Junit Integration Test**

In this lab, we will work with a spring boot microservice that can service the following REST API requests: GET, POST, PUT DELETE API requests to perform CRUD (Create Read Update Delete) operations.

We will test the system as a black box and test the system from outside. We will test for a REST resource and test the following:

- The HTTP response code
- The HTTP response headers
- The payload (JSON)

Open a File explorer to navigate to the directory

### **C:\apitestlabs\customerservice**

This directory contains a simple spring boot customer microservice that handles the following requests:

GET request to get all customers

URL: <http://localhost:8090/api/customers>

GET request to get a customer using customer id

URL: <http://localhost:8090/api/customers/{id}>

id is the template variable

For example <http://localhost:8090/api/customers/2> will return customer data for a customer id value 2

POST request to create a new customer

URL: <http://localhost:8090/api/customers>

Post a Json data via request body

PUT request to update an existing customer with an id

URL: <http://localhost:8090/api/customers/{id}>

id is the template variable

For example <http://localhost:8090/api/customers/2>

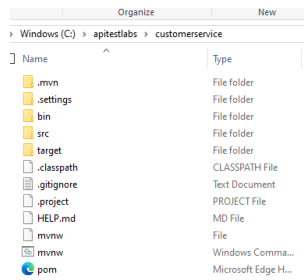
DELETE request to delete an existing customer with an id

URL: <http://localhost:8090/api/customers/{id}>

id is the template variable

For example <http://localhost:8090/api/customers/2>

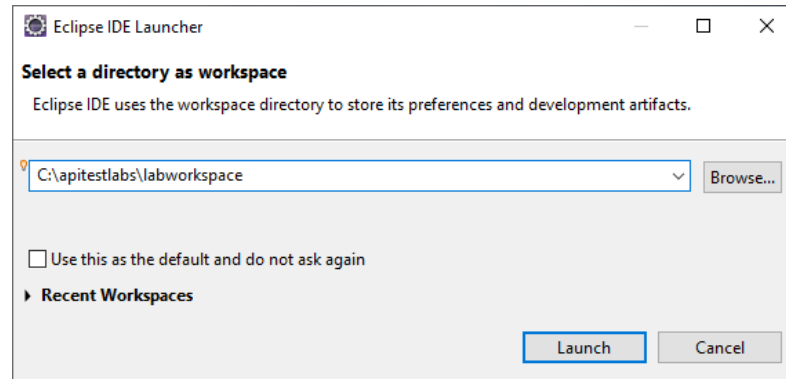
## 1. Open a File Explorer, Navigate to **C:\apitestlabs\customerservice**



Review the resources in this microservice directory.

You may open this project in Eclipse and review the source code and configuration.

## 2. Start your eclipse and go to your lab workspace: **C:\apitestlabs\labworkspace**



## 3. Let us import the customer service project. In Eclipse,

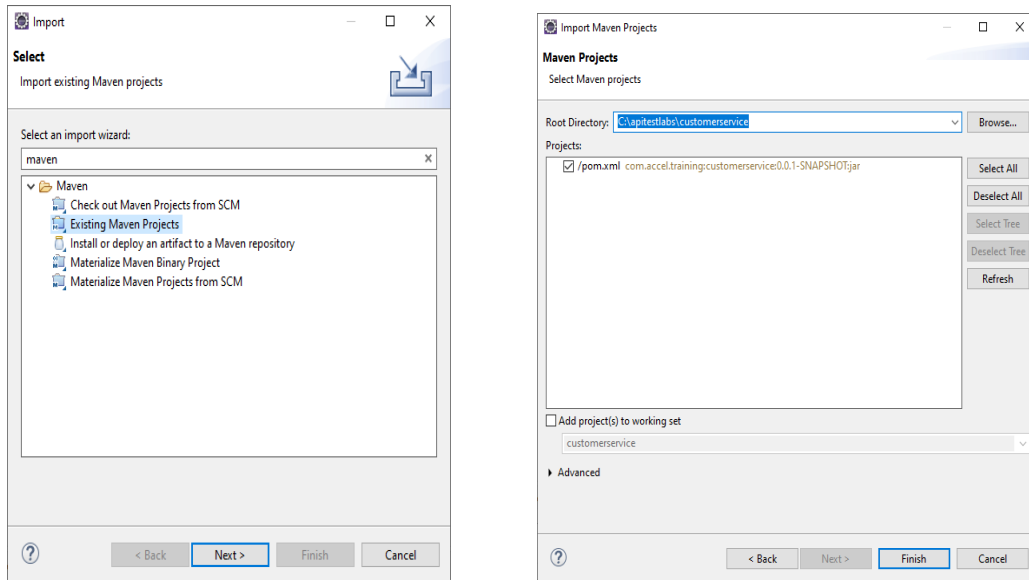
Click on **File → Import → Maven → Existing maven projects**

Click on **Next**.

Navigate to **C:\apitestlabs\customerservice**

Select this project.

Click on **Finish**.



4. In Eclipse Project Explorer view, Select the project → right click on customer service project → Maven → Update Project

In Project Explorer view, click on **src/test/java** → **com.webage.tests** → **CustomerAPIJUnitIntegrationTest.java** and open this source code.

```

labworkspace - customerservice/src/test/java/com/webage/tests/CustomerAPIJUnitIntegrationTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

CustomerAPIJUnitIntegrationTest.java X
1 import static org.assertj.core.api.Assertions.assertThat;
2 import static org.junit.jupiter.api.Assertions.assertTrue;
3
4 import java.net.URI;
5 import java.net.http.HttpClient;
6 import java.net.http.HttpRequest;
7 import java.net.http.HttpResponse;
8 import java.net.http.HttpResponse.BodyHandlers;
9 import java.util.Optional;
10 import org.junit.jupiter.api.DisplayName;
11 import org.junit.jupiter.api.Test;
12 import org.junit.jupiter.api.Test;
13
14 public class CustomerAPIJUnitIntegrationTest {
15
16     //Test
17     @DisplayName("Ensures that the api returns OK status code 200")
18     public void ensureThatUserAPICallReturnsStatusCode200() throws Exception {
19         HttpClient client = HttpClient.newBuilder().build();
20         HttpRequest request = HttpRequest.newBuilder().uri(URI.create("http://localhost:8090/api/customers")).build();
21         HttpResponse<String> response = client.send(request, BodyHandlers.ofString());
22         assertThat(response.statusCode()).isEqualTo(200);
23     }
24
25     //Test
26     @DisplayName("Ensures that the content type starts with application/json")
27     void ensureThatJsonIsReturnedAsContentType() throws Exception {
28         HttpClient client = HttpClient.newBuilder().build();
29         HttpRequest request = HttpRequest.newBuilder().uri(URI.create("http://localhost:8090/api/customers/1")).build();
30         HttpResponse<String> response = client.send(request, BodyHandlers.ofString());
31         Optional<String> firstValue = response.headers().firstValue("Content-Type");
32         String string = firstValue.get();
33         assertThat(string).startsWith("application/json");
34     }
35
36     //Test
37     @DisplayName("Ensure that the JSON for the user id 1 contains a reference to steve")
38     void ensureJsonContainsCustomerName() throws Exception {
39         HttpClient client = HttpClient.newBuilder().build();
40         HttpRequest request = HttpRequest.newBuilder().uri(URI.create("http://localhost:8090/api/customers/1")).build();
41         HttpResponse<String> response = client.send(request, BodyHandlers.ofString());
42         String body = response.body();
43         // For easy to see the output
44         System.out.println(body);
45         assertTrue(body.contains("name\":\"steve\""));
46     }
47
48 }
49
50

```

5. Review the Test code in this file.

6. Let us test our API GET request and returns the status code 200 OK..  
Uncomment the `@Test` annotation above the following method:  
**ensureThatUserAPICallReturnStatusCode200 ()**

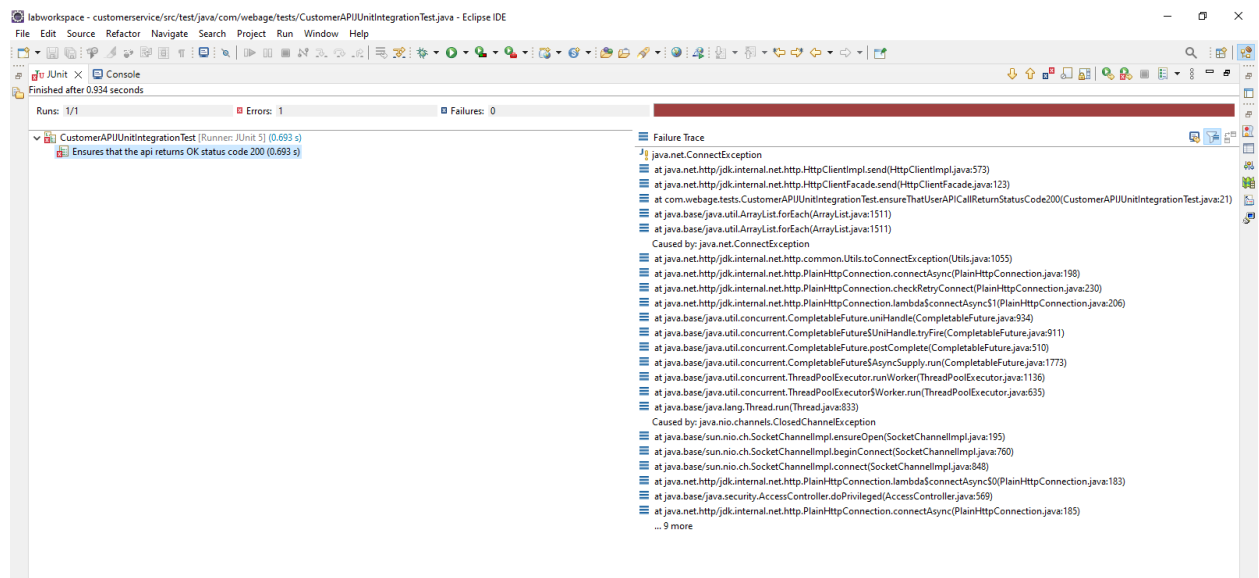
```
@Test
@DisplayName("Ensures that the api returns OK status code 200")
public void ensureThatUserAPICallReturnStatusCode200() throws Exception {
    HttpClient client = HttpClient.newBuilder().build();
    HttpRequest request =
    HttpRequest.newBuilder().uri(URI.create("http://localhost:8090/api/customers"))
    .build();
    HttpResponse<String> response = client.send(request,
    BodyHandlers.ofString());
    assertThat(response.statusCode()).isEqualTo(200);
}
```

and save the file.

7. Select the method **ensureThatUserAPICallReturnStatusCode200()** → right click → **Run As** → **Junit Test**

and run the Junit Test.

What is the result? Pass | Fail ?



If the API application is not running, this test should fail as shown in the screen shot.

8. Let us start and run the application. Click on **src/main/java ---> com.webage -> CustomerServiceApplication.java -> right click --> Run As -> Java Application**

```

2024-03-16 10:40:24.686 INFO 12692 --- [main] com.webage.CustomerServiceApplication : Starting CustomerServiceApplication on DESKTOP-993BA2R with PID 12692 (C:\apitestlabs\customerservice
2024-03-16 10:40:24.611 INFO 12692 --- [main] com.webage.CustomerServiceApplication : No active profile set, falling back to default profiles: default
2024-03-16 10:40:25.684 INFO 12692 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2024-03-16 10:40:25.792 INFO 12692 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 92ms. Found 1 JPA repository interfaces.
2024-03-16 10:40:26.986 INFO 12692 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8090 (http)
2024-03-16 10:40:27.001 INFO 12692 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2024-03-16 10:40:27.001 INFO 12692 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.33]
2024-03-16 10:40:27.175 INFO 12692 --- [main] o.a.c.c.c.[Tomcat].[/api] : Initializing Spring embedded WebApplicationContext
2024-03-16 10:40:27.177 INFO 12692 --- [main] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed in 2480 ms
2024-03-16 10:40:27.344 INFO 12692 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2024-03-16 10:40:27.356 WARN 12692 --- [main] com.zaxxer.hikari.util.DriverDataSource : Registered driver with driverClassName=org.hsqldb.jdbcDriver was not found, trying direct instantiation
2024-03-16 10:40:27.697 INFO 12692 --- [main] com.zaxxer.hikari.pool.PoolBase : HikariPool-1 - Driver does not support get/set network timeout for connections. (feature not supported)
2024-03-16 10:40:27.782 INFO 12692 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2024-03-16 10:40:27.888 INFO 12692 --- [main] o.hibernate.jpa.internal.util.LogHelper : MH0000204: Processing PersistenceUnitInfo [name: default]
2024-03-16 10:40:28.059 INFO 12692 --- [main] org.hibernate.Version : MH0000412: Hibernate ORM core version 5.4.12.Final
2024-03-16 10:40:28.302 INFO 12692 --- [main] o.hibernate.annotations.common.Version : HCANN000001: Hibernate Commons Annotations {5.1.0.Final}
2024-03-16 10:40:28.529 INFO 12692 --- [main] org.hibernate.dialect.Dialect : MH0000400: Using dialect: org.hibernate.dialect.HSQLDialect
2024-03-16 10:40:29.571 INFO 12692 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : MH0000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.
2024-03-16 10:40:29.590 INFO 12692 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2024-03-16 10:40:30.061 WARN 12692 --- [main] jpahibernate.Configuration$JpaMethodConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during vi
2024-03-16 10:40:30.343 INFO 12692 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2024-03-16 10:40:30.718 INFO 12692 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8090 (http) with context path '/api'
2024-03-16 10:40:30.723 INFO 12692 --- [main] com.webage.CustomerServiceApplication : Started CustomerServiceApplication in 7.267 seconds (JVM running for 8.056)
  
```

Review the application log messages.

9. Select the method **ensureThatUserAPICallReturnStatusCode200()** → right click → Run As → Junit Test

and run the Junit Test.

What is the result? Pass | Fail ?

```

13
14 public class CustomerAPIUnitIntegrationTest {
15
16     @Test
17     @DisplayName("Ensures that the api returns OK status code 200")
18     public void ensureThatUserAPICallReturnStatusCode200() throws Exception {
19         HttpClient client = HttpClient.newBuilder().build();
20         HttpRequest request = HttpRequest.newBuilder().uri(URI.create("http://localhost:8090/api/customers")).build();
21         HttpResponse<String> response = client.send(request, BodyHandlers.ofString());
22         assertThat(response.statusCode(), isEqualTo(200));
23     }
24
25
  
```

JUnit Console: Finished after 1.616 seconds. Runs: 1/1, Errors: 0, Failures: 0. CustomerAPIUnitIntegrationTest [Runner: JUnit 5] (1.446 s). ensureThatUserAPICallReturnStatusCode200 (1.446 s).

10. Check the console view for any log messages.

11. Let us test that the API call returns the JSON type data content. The method **ensurethatJsonIsReturnedAsContentType()** contains code to test this task. Uncomment the `@Test` annotation above this method and **save the file**.

```
@Test
@DisplayName("Ensures that the content type starts with application/json")
void ensureThatJsonIsReturnedAsContentType() throws Exception {
    HttpClient client = HttpClient.newBuilder().build();
    HttpRequest request =
HttpRequest.newBuilder().uri(URI.create("http://localhost:8090/api/customers/1")).build();

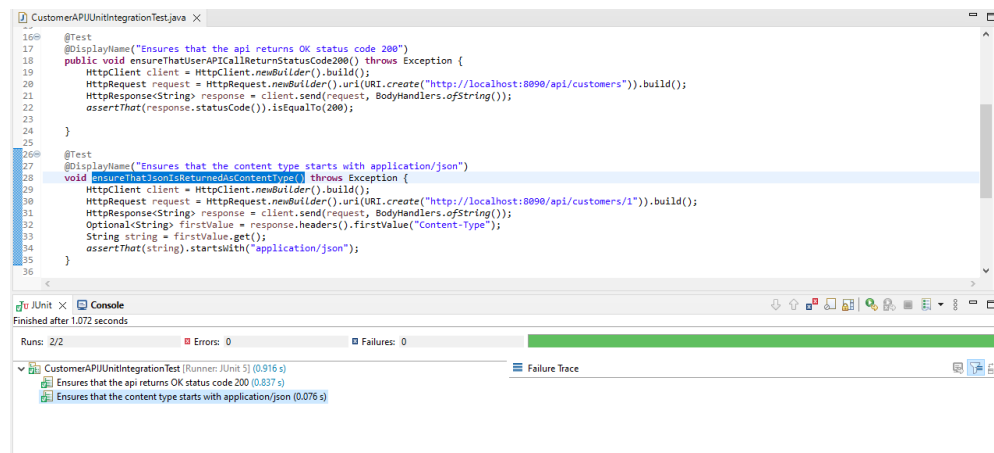
    HttpResponse<String> response = client.send(request,
BodyHandlers.ofString());
    Optional<String> firstValue = response.headers().firstValue("Content-Type");
    String string = firstValue.get();
    assertThat(string).startsWith("application/json");
}
```

12. Select the method → right click → Run As → Junit Test

and run the Junit Test.

What is the result? Pass | Fail ?

Review the **console** messages.



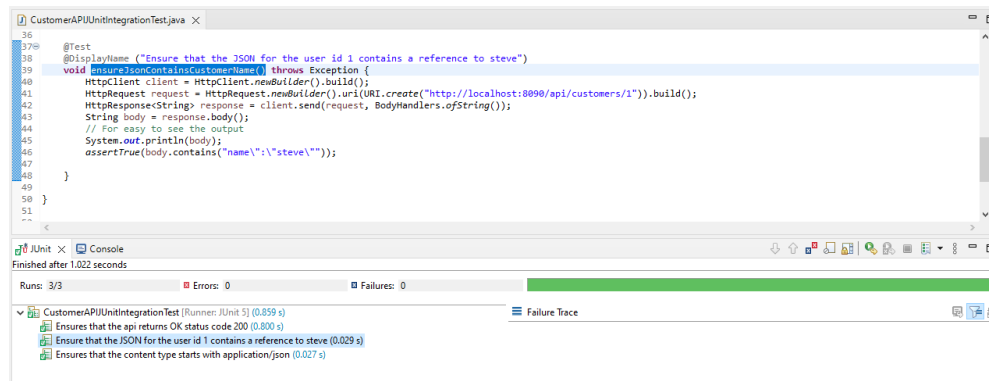
13. Let us test API returns a data that contains a specific value. For example, for the customer with id=1, name must be steve. Let us test that API returns a name steve for customer id=1.

Uncomment the `@Test` annotation above the method **ensureJsonContainsCustomerName()**

```
@Test
@DisplayName ("Ensure that the JSON for the user id 1 contains a reference to
steve")
void ensureJsonContainsCustomerName() throws Exception {
    HttpClient client = HttpClient.newBuilder().build();
    HttpRequest request =
HttpRequest.newBuilder().uri(URI.create("http://localhost:8090/api/customers/1")).build();
    HttpResponse<String> response = client.send(request,
BodyHandlers.ofString());
    String body = response.body();
    // For easy to see the output
    System.out.println(body);
    assertTrue(body.contains("name\":\"steve\""));
}
```

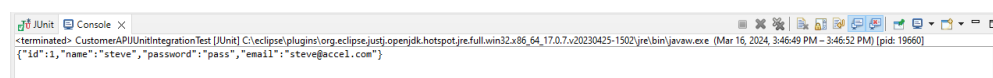
and save the file.

14. Run the test. Select the method → right click → Run As → Junit Test and run the Junit Test.



What is the result? Pass | Fail ?

Review the **console** messages.



15. Based on what you have learned, write your own methods and tests.