

Spring Boot Microservices API Test using Postman

In this lab, Let us test a spring boot microservice that can service the following API requests: GET, POST, PUT DELETE API requests to perform CRUD (Create Read Update Delete) operations.

How to test the API end points with HTTP requests?

Use Postman !

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>

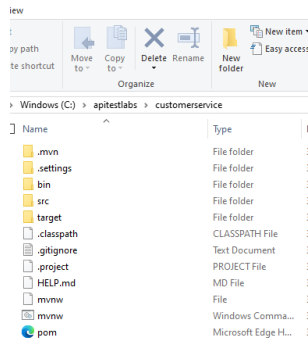
DELETE request to delete an existing customer with an id

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**

cd C:\apitestlabs\customerservice

```
customer service
Microsoft Windows [Version 10.0.19045.4170]
(c) Microsoft Corporation. All rights reserved.

C:\Users\saravanan>title customer service
C:\Users\saravanan>cd C:\apitestlabs\customerservice
C:\apitestlabs\customerservice>
```

3. Let us build the service using maven. Type the following command in the command window

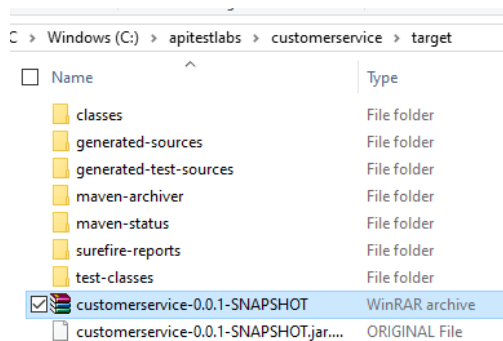
mvn install

```
2024-03-16 10:05:22.770 INFO 10000 --- [EX-3H0U0M0R00K] [com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown completed.
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] --- maven-jar-plugin:3.1.2:jar (default-jar) @ customerservice ---
[INFO] Building jar: C:\apitestlabs\customerservice\target\customerservice-0.0.1-SNAPSHOT.jar
[INFO] --- spring-boot-maven-plugin:2.2.6.RELEASE:repackage (repackage) @ customerservice ---
[INFO] Replacing main artifact with repackaged archive
[INFO] --- maven-install-plugin:2.5.2:install (default-install) @ customerservice ---
[INFO] Installing C:\apitestlabs\customerservice\target\customerservice-0.0.1-SNAPSHOT.jar to C:\Users\saravanan\.m2\repository\com\accel\training\customerservice\0.0.1-SNAPSHOT\customerservice-0.0.1-SNAPSHOT.jar
[INFO] Installing C:\apitestlabs\customerservice\pom.xml to C:\Users\saravanan\.m2\repository\com\accel\training\customerservice\0.0.1-SNAPSHOT\customerservice-0.0.1-SNAPSHOT.pom
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 19.809 s
[INFO] Finished at: 2024-03-16T09:51:25-04:00
[INFO]
C:\apitestlabs\customerservice>
```

Make sure you get the Build Successful message.

4. Use the file explorer to navigate to **C:\apitestlabs\customerservice\target**

What is in this directory? Can you view the application jar file as shown in the following screen shot?



This is the spring boot application that can be executed using the java command.

5. In the Command window, type the command change directory to **C:\apitestlabs\customerservice\target**

cd target

```
customer service
C:\apitestlabs\customerservice>cd C:\apitestlabs\customerservice\target
C:\apitestlabs\customerservice\target>dir *.jar
Volume in drive C is Windows
Volume Serial Number is 1255-38EE

Directory of C:\apitestlabs\customerservice\target

03/16/2024  09:51 AM          38,098,507 customerservice-0.0.1-SNAPSHOT.jar
               1 File(s)          38,098,507 bytes
               0 Dir(s)  36,862,844,928 bytes free

C:\apitestlabs\customerservice\target>
```

6. Run this application using the following command

java -jar customerservice-0.0.1-SNAPSHOT.jar

```

customer service -java -jar customerservice-0.0.1-SNAPSHOT.jar
C:\apitestlabs\customerservice\target>java -jar customerservice-0.0.1-SNAPSHOT.jar

:: Spring Boot ::
(v2.2.6.RELEASE)

2024-03-16 09:55:36.899 INFO 18628 --- [main] com.webgate.CustomerServiceApplication : Starting CustomerServiceApplication v0.0.1-SNAPSHOT on DESKTOP-9938A2R with PID 18628 (C:\apitestlabs\customerservice\target\customerservice-0.0.1-SNAPSHOT.jar started by saravanan in C:\apitestlabs\customerservice\target)
2024-03-16 09:55:36.817 INFO 18628 --- [main] com.webgate.CustomerServiceApplication : No active profile set, falling back to default profiles: default
2024-03-16 09:55:38.004 INFO 18628 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2024-03-16 09:55:38.110 INFO 18628 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 88ms. Found 1 JPA repository interfaces.
2024-03-16 09:55:39.358 INFO 18628 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8090 (http)
2024-03-16 09:55:39.374 INFO 18628 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2024-03-16 09:55:39.375 INFO 18628 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.33]
2024-03-16 09:55:39.486 INFO 18628 --- [main] o.a.c.c.c.[Tomcat].[localhost].[/api] : Initializing Spring embedded WebApplicationContext
2024-03-16 09:55:39.487 INFO 18628 --- [main] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed in 2559 ms
2024-03-16 09:55:39.651 INFO 18628 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2024-03-16 09:55:39.666 WARN 18628 --- [main] com.zaxxer.hikari.util.DriverDataSource : Registered driver with driverClassName=org.hsqldb.jdbcDriver was not found, trying direct instantiation.
2024-03-16 09:55:40.049 INFO 18628 --- [main] com.zaxxer.hikari.pool.PoolBase : HikariPool-1 - Driver does not support get/set network timeout for connections. (feature not supported)
2024-03-16 09:55:40.053 INFO 18628 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2024-03-16 09:55:40.255 INFO 18628 --- [main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]
2024-03-16 09:55:40.398 INFO 18628 --- [main] org.hibernate.Version : HHH0000412: Hibernate ORM core version 5.4.12.Final
2024-03-16 09:55:40.658 INFO 18628 --- [main] o.hibernate.annotations.common.Version : HCANN000001: Hibernate Commons Annotations {5.1.0.Final}
2024-03-16 09:55:40.936 INFO 18628 --- [main] org.hibernate.dialect.Dialect : HHH000400: Using dialect: org.hibernate.dialect.HSQLDialect
2024-03-16 09:55:41.777 INFO 18628 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
2024-03-16 09:55:41.786 INFO 18628 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2024-03-16 09:55:42.248 WARN 18628 --- [main] jpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
2024-03-16 09:55:42.468 INFO 18628 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2024-03-16 09:55:42.883 INFO 18628 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8090 (http) with context path '/api'
2024-03-16 09:55:42.887 INFO 18628 --- [main] com.webgate.CustomerServiceApplication : Started CustomerServiceApplication in 6.951 seconds (JVM running for 7.754)

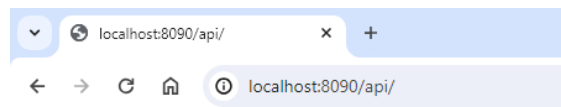
```

This application launches a spring boot microservice listening at port 8090.

Review the application log messages.

- Let us use a web browser and test the following end points. Open browser and go to the following URLs one by one and test.

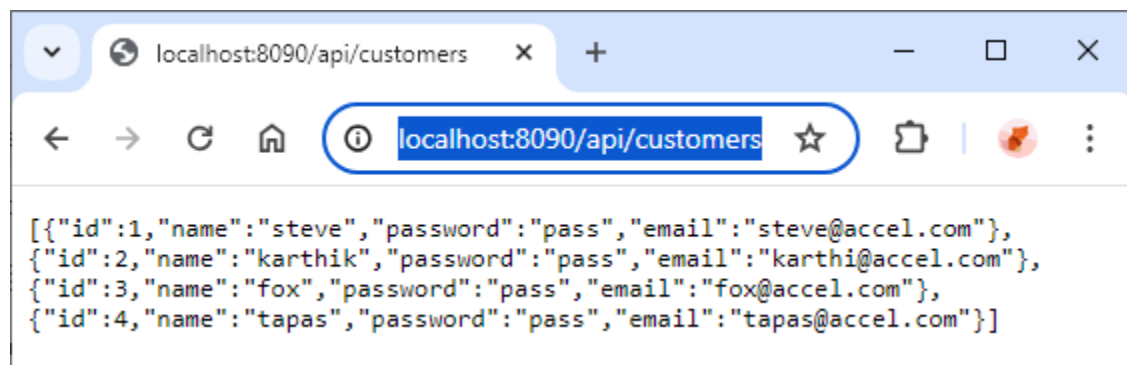
<http://localhost:8090/api>



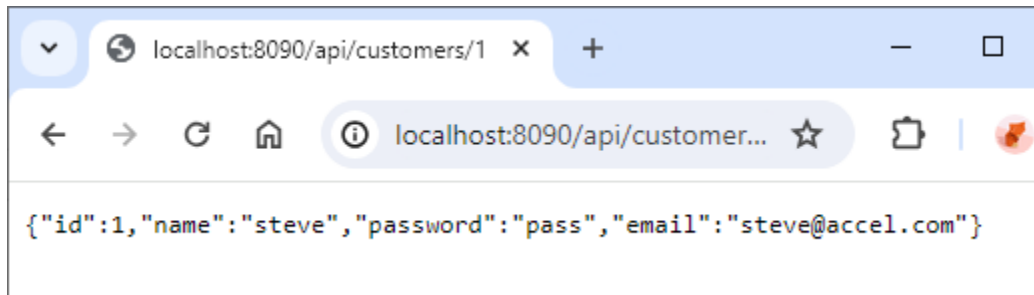
The Customer API service is up and running!

Instance: -744644827,
Date/Time: 3/15/24, 8:36:01 PM Eastern Daylight Time
CallCount: 1

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

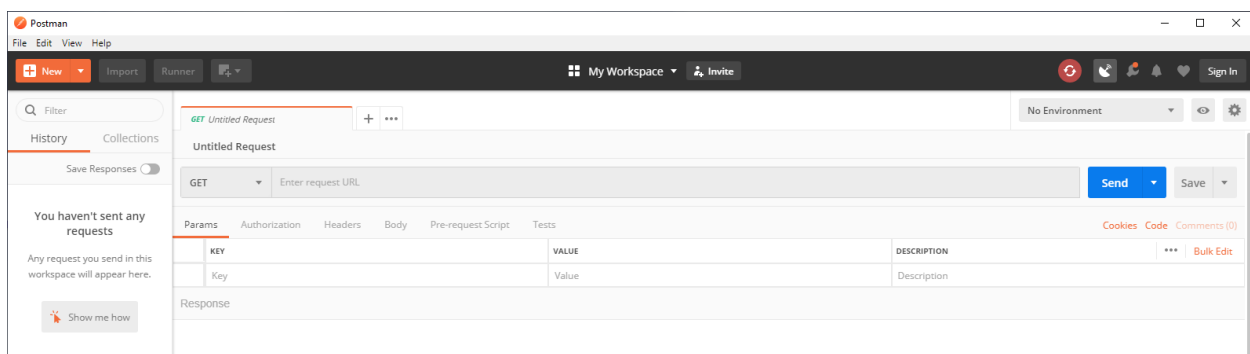


<http://localhost:8090/api/customers/1>



Part 2 Use Postman to REST end points and CRUD Operations

8. Launch the postman application. Click on the postman in your machine desktop to launch a postman instance.



9. Let us create a new GET request to send a request to this URL

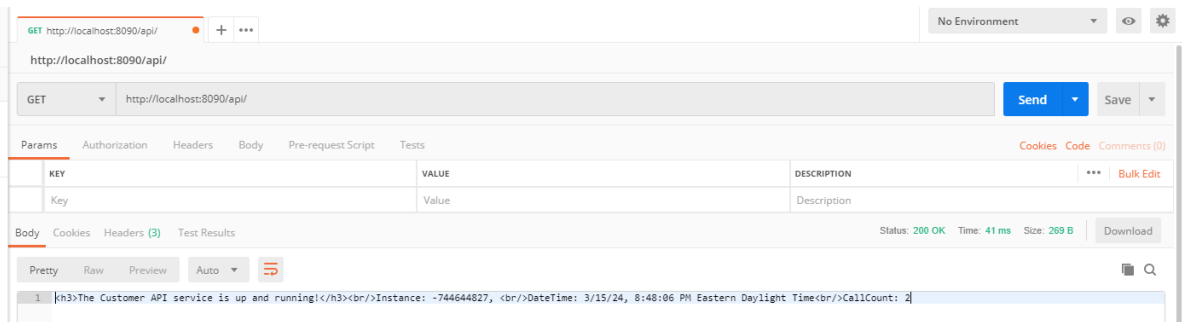
<http://localhost:8090/api/>

In Postman's New request, use the following data values and then click **send**.

Method: GET

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

Click on **send** a few times.



Can you see the API call count in the response body?

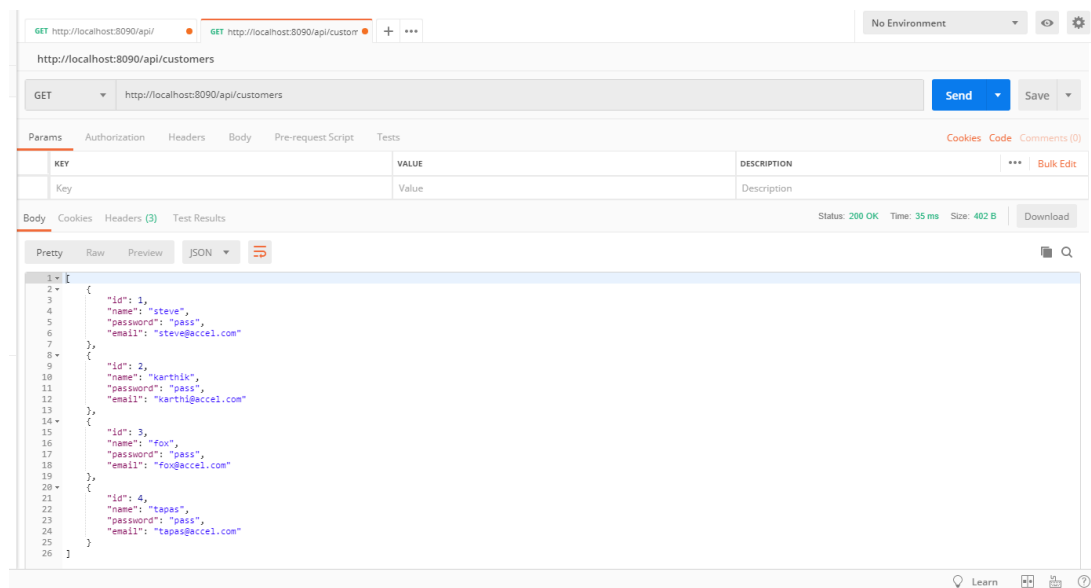
10. Let us get all customers. Create a new postman request to this URL

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

Method: **GET**

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

Click on **send**. Can you get the list of customers from the API?



11. We want to create a postman collection with all requests. Let us save every request under test to this collection and call the collection as **“Customers API Tests.”** Click on the **Save** button (next to Send) and **save as**



Use the following details and save this request to a collection:

Customers API Tests

Request name

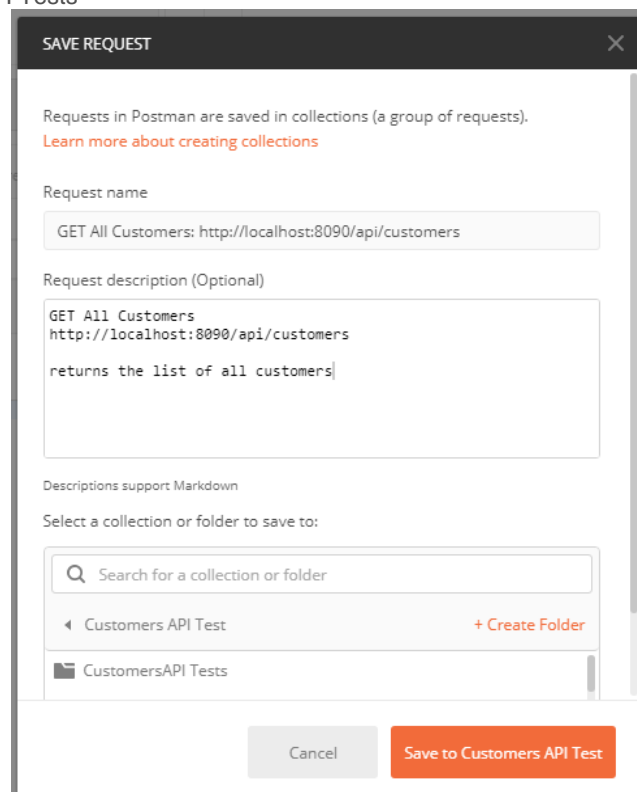
GET All Customers: <http://localhost:8090/api/customers>

Request description (Optional)

GET All Customers
<http://localhost:8090/api/customers>
returns the list of all customers

Select a collection or folder to save to:

Customers API Tests

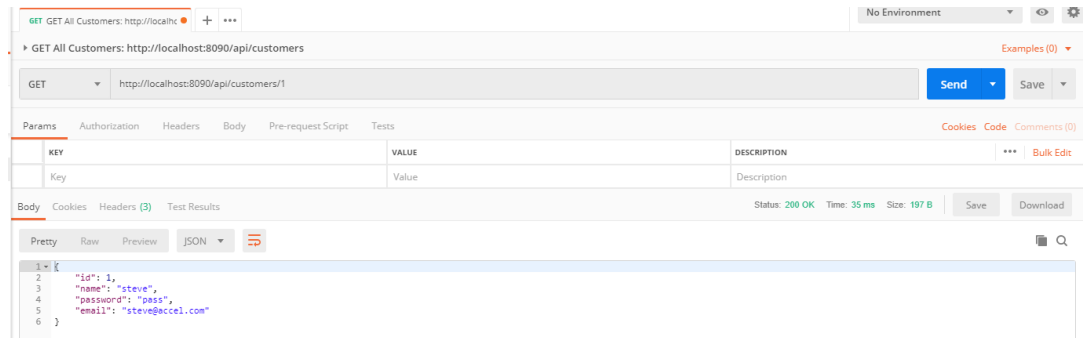


12. Let us get a customer data using customer id. Create a new postman request with the following request data:

Method: **GET**

URL: <http://localhost:8090/api/customers/1>

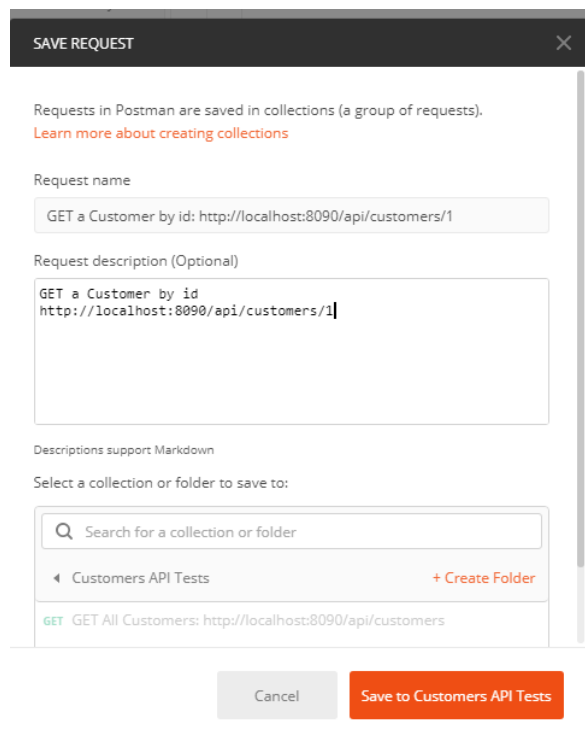
Click on **send**.



Inspect the response Body, Headers, Status, Size, and any other details.

What is the HTTP status code?

13. Save this request to **Customers API Tests** collection.



14. Let us create a new customer and post the data to API. Create a new postman request with following details.

Method: **POST**

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

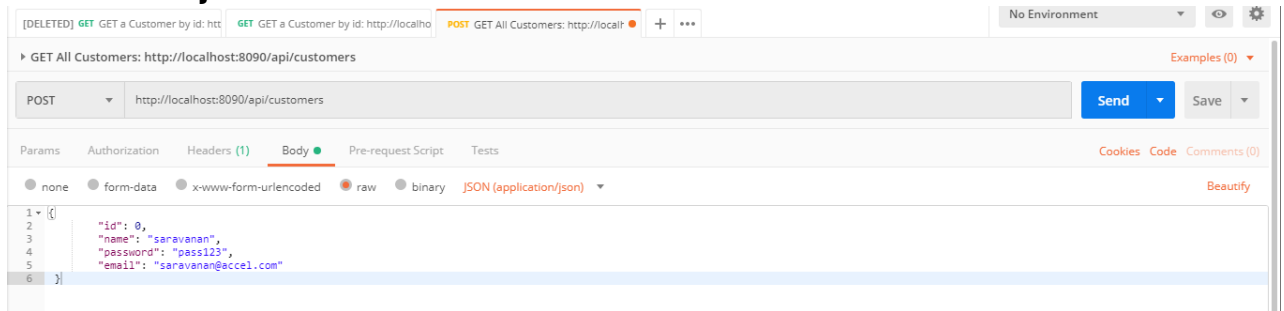
Request Body:

raw

JSON (application/json)

Pay load data:

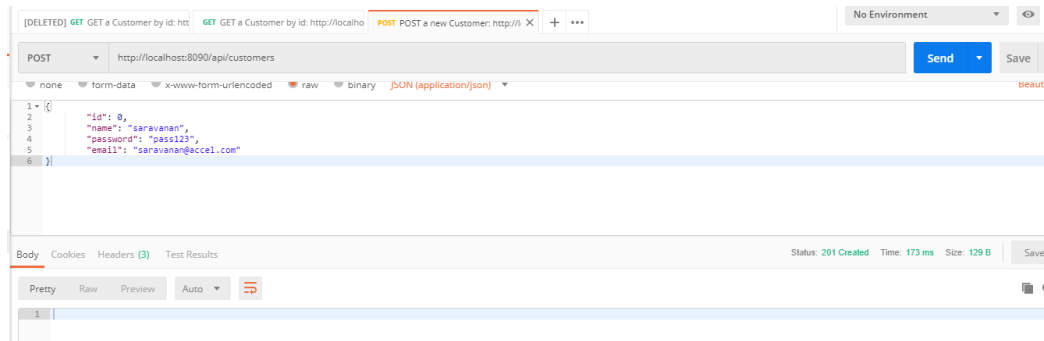
```
{  
  "id": 0,  
  "name": "saravanan",  
  "password": "pass123",  
  "email": "saravanan@accel.com"  
}
```



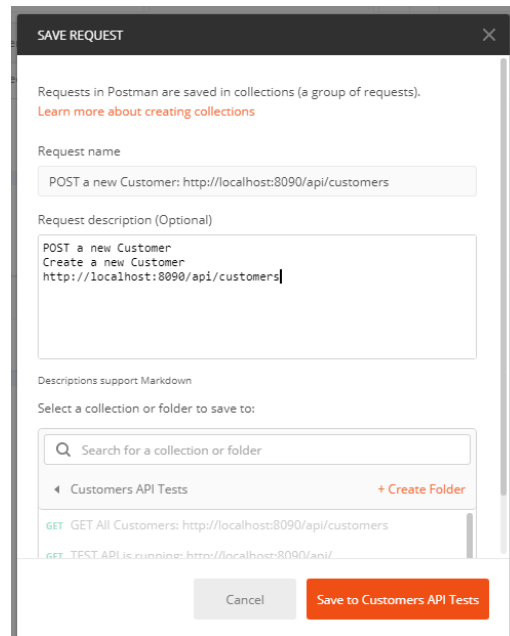
Click on **send**. What is the response status code? **201 Created?**

Where is the new Customer created?

How to find the location of this customer? Click on response Headers to see this.



15. Save the previous POST request to **Customers API Tests** collection.



16. Let us test the newly created Customer data using GET requests. Create new postman GET requests to the following URL and send.

Method: **GET**

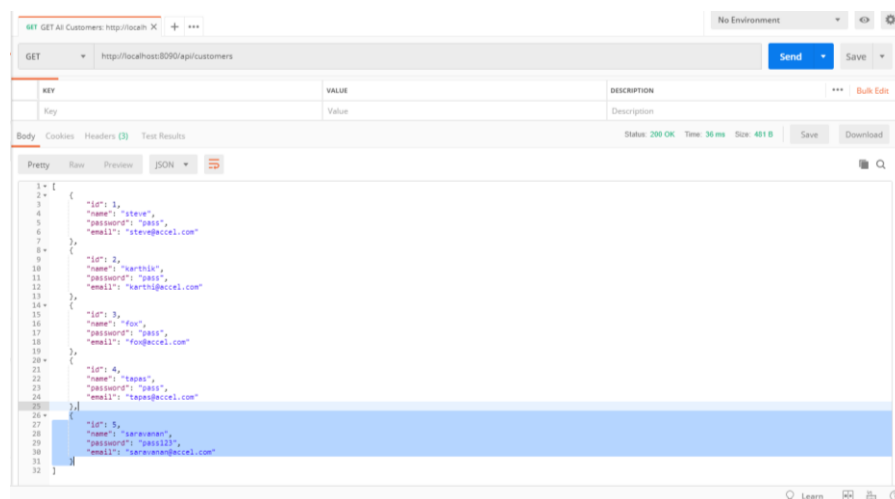
URL: <http://localhost:8090/api/customers>

Method: **GET**

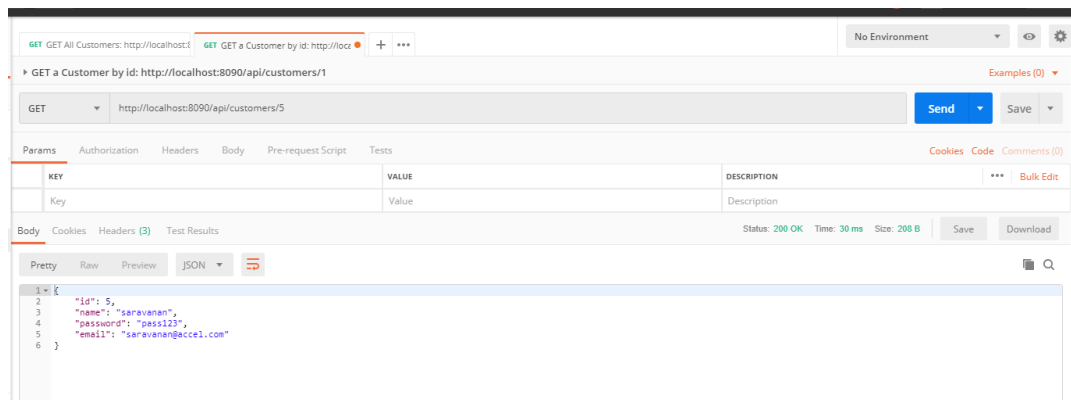
URL: <http://localhost:8090/api/customers/5>

Can you get the new Customer data?

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



<http://localhost:8090/api/customers/5>



17. Let us update the new Customer data. We will use PUT request with a request body data with modified password and email data. Create a new postman PUT request with the following data.

Method: **PUT**

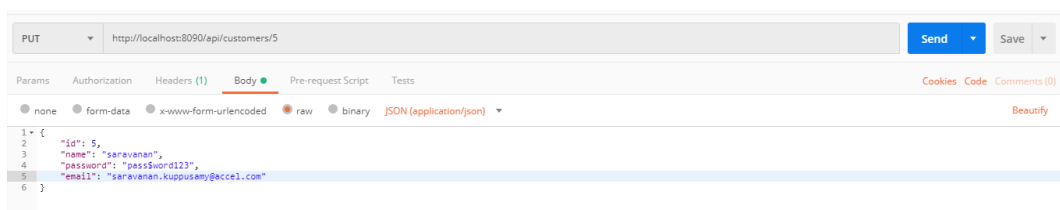
URL: <http://localhost:8090/api/customers/5>

Request Body:

raw **JSON (application/json)**

Pay load data:

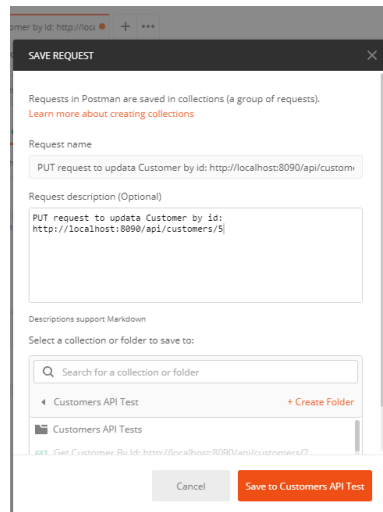
```
{
  "id": 5,
  "name": "saravanan",
  "password": "pass$word123",
  "email": "saravanan.kuppusmy@accel.com"
}
```



Click on **send**.

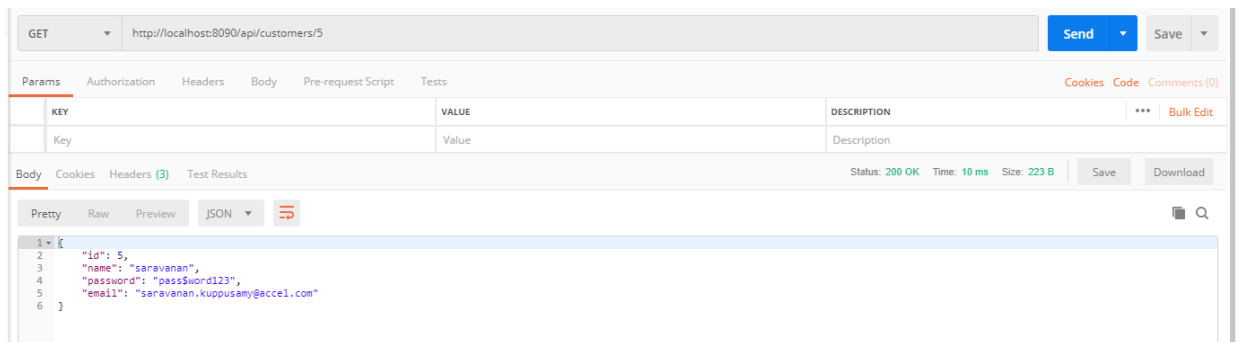
What is the response code?

18. Save the previous PUT request to **Customers API Tests** collection



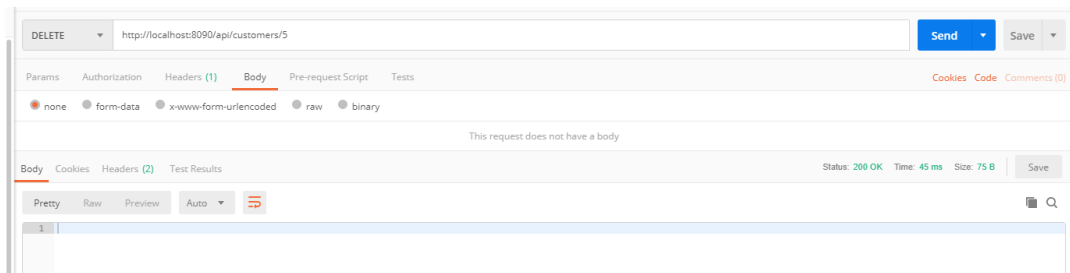
19. Let us verify the updated Customer data using GET requests. Use the following end points and GET method invocations to verify the modified customer data.

<http://localhost:8090/api/customers>
<http://localhost:8090/api/customers/5>



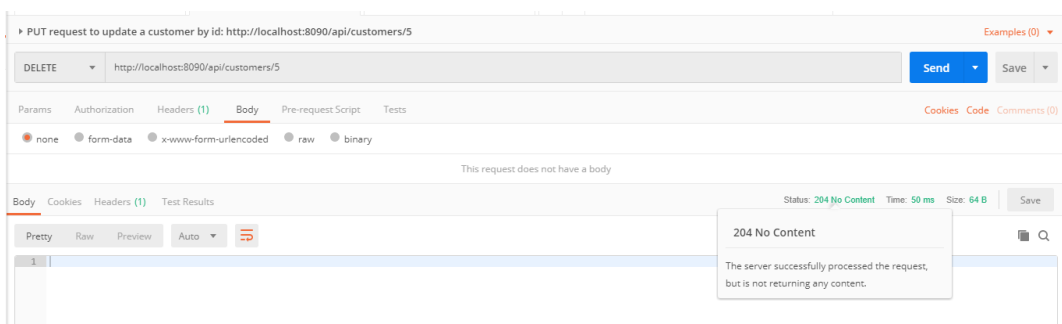
20. Let us DELETE an existing Customer using the customer id. Create a new postman DELETE request with the following data.

Method: **DELETE**
URL: <http://localhost:8090/api/customers/5>
Body: none

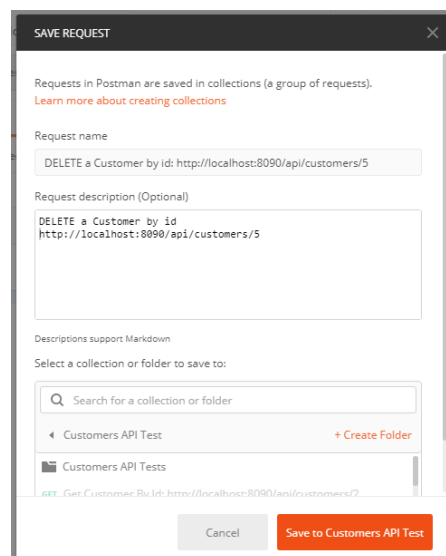


Click on **send**.

What is the response?



21. Save the previous DELETE request to **Customers API Tests** collection.



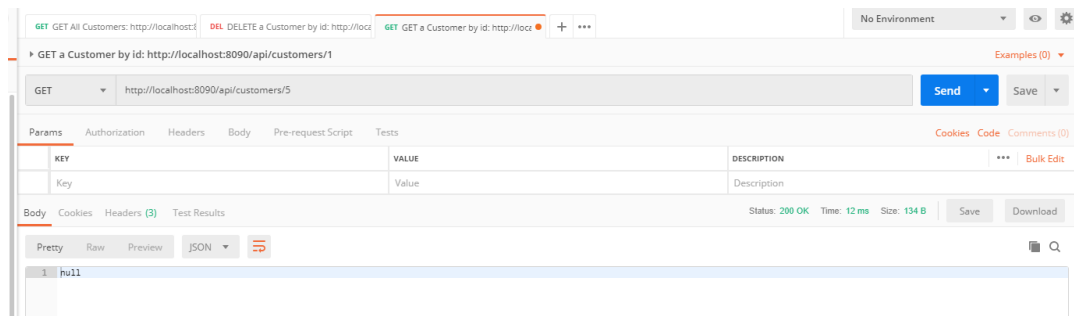
22. Now, let us use the GET requests to verify that customer data is no longer available. Use the following URLs to see postman responses

Method: **GET**

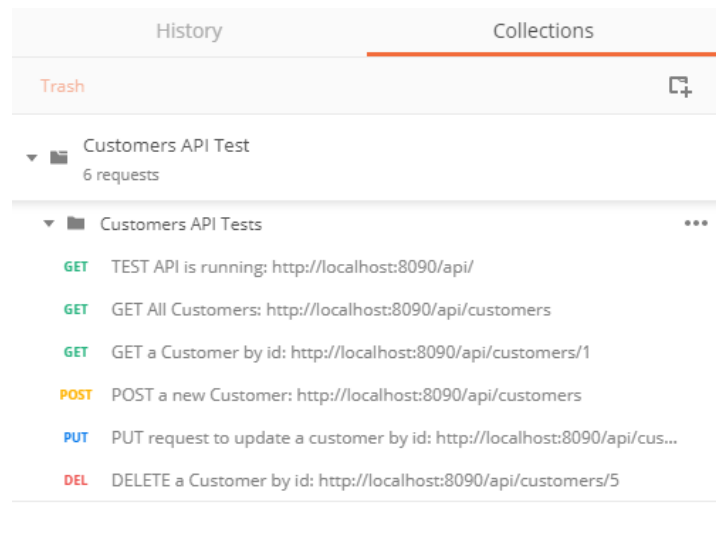
URL: <http://localhost:8090/api/customers>

Method: **GET**

URL: <http://localhost:8090/api/customers/5>

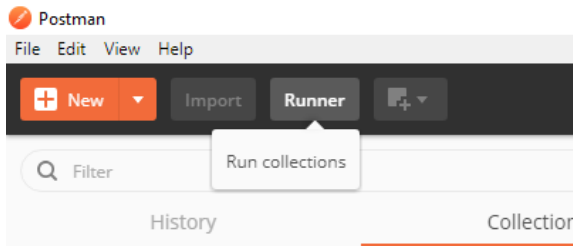


23. Review the postman collection: **Customers API Tests**. Run the requests with new data, modify the data and delete and test API.

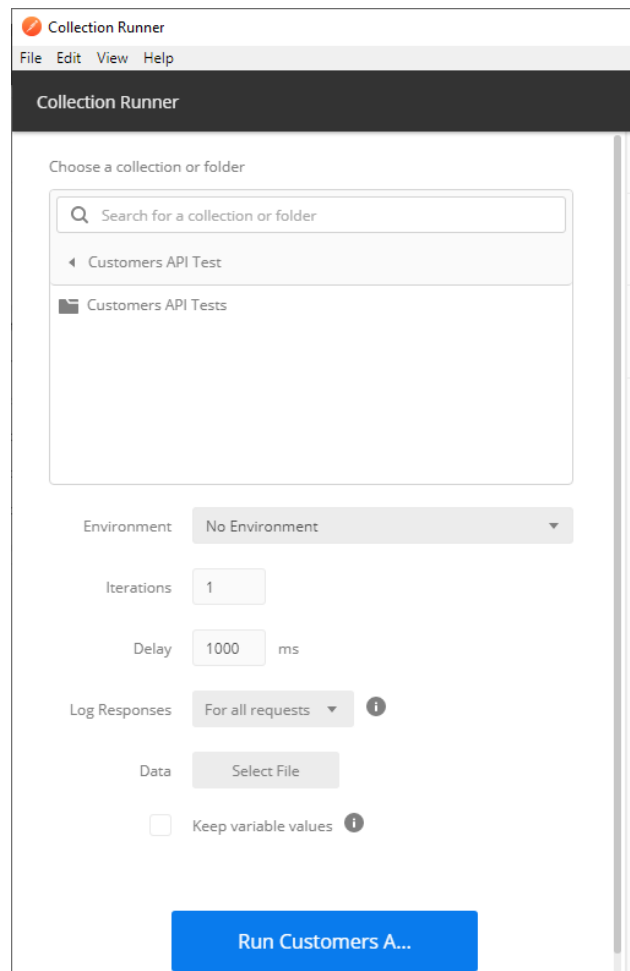


Part 3 Run the postman collection

24. Let us run the postman collection: **Customers API Tests**. In postman, click on **Runner → Run Collections**



25. In the collection runner dialog, use the following configuration with a delay of **1000 ms**, all requests log responses and **Run Customers API Tests collection**.



26. See the results and review the results.

The screenshot shows the Postman Collection Runner interface for a collection named 'Customers API Test'. The interface has a dark header with 'Collection Runner' and 'Run Results' tabs. Below the header, there are two circular progress indicators: a green one with '0 PASSED' and a red one with '0 FAILED'. The test name 'Customers API Test' is displayed, along with 'No Environment' and 'just now'. Buttons for 'Run Summary', 'Export Results', 'Retry', and 'New' are visible. The main area shows a list of test iterations. The first iteration is expanded, showing a table of test results. The table has columns for the test name, status, and response details. The tests are: 'GET TEST API is running: http://localhost:8090/api/' (200 OK, 14 ms, 153 B), 'GET GET All Customers: http://localhost:8090/api/customers' (200 OK, 12 ms, 445 B), 'GET GET a Customer by id: http://localhost:8090/api/customers/1' (200 OK, 11 ms, 67 B), 'POST POST a new Customer: http://localhost:8090/api/customers' (201 Created, 10 ms, 0 B), 'PUT PUT request to update a customer by id: http://localhost:8090/api/customers/5' (200 OK, 9 ms, 0 B), and 'DELETE DELETE a Customer by id: http://localhost:8090/api/customers/5' (500 Internal Server Error, 17 ms, 188 B).

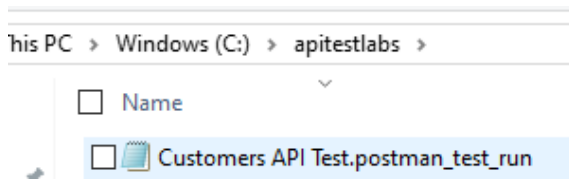
Test Name	Status	Response Details
GET TEST API is running: http://localhost:8090/api/	200 OK	14 ms, 153 B
GET GET All Customers: http://localhost:8090/api/customers	200 OK	12 ms, 445 B
GET GET a Customer by id: http://localhost:8090/api/customers/1	200 OK	11 ms, 67 B
POST POST a new Customer: http://localhost:8090/api/customers	201 Created	10 ms, 0 B
PUT PUT request to update a customer by id: http://localhost:8090/api/customers/5	200 OK	9 ms, 0 B
DELETE DELETE a Customer by id: http://localhost:8090/api/customers/5	500 Internal Server Error	17 ms, 188 B

27. Click on **Run Summary** and review.

The screenshot shows the Postman Collection Runner interface for the 'Customers API Test' collection, now displaying the 'Run Summary' tab. The header shows 'Collection Runner', 'Run Results', and 'Run Summary' tabs. The test name 'Customers API Test' is displayed, along with 'No Environment' and 'a min ago'. Buttons for 'Export Results' and 'New' are visible. The main area shows a list of test results. The table has columns for the test name, status, and response details. The tests are: 'GET TEST API is running: http://localhost:8090/a...', 'GET GET All Customers: http://localhost:8090/a...', 'GET GET a Customer by id: http://localhost:809...', 'POST POST a new Customer: http://localhost:809...', 'PUT PUT request to update a customer by id: ht...', and 'DELETE DELETE a Customer by id: http://localhost...'. All tests are marked with a green checkmark, indicating they passed.

Test Name	Status	Response Details
GET TEST API is running: http://localhost:8090/a...	✓	
GET GET All Customers: http://localhost:8090/a...	✓	
GET GET a Customer by id: http://localhost:809...	✓	
POST POST a new Customer: http://localhost:809...	✓	
PUT PUT request to update a customer by id: ht...	✓	
DELETE DELETE a Customer by id: http://localhost...	✓	

28. Click on **Export Results** and export the results to **C:\apitestlabs** directory.



By default it will export the results to a file called
Customers API Test.postman_test_run.json

29. Use a text editor to open the file and review the file contents.

