**Customer Service spring boot restful web service application:**

The main purpose of this Customer Service spring boot restful web service application is to demonstrate the functionalities like adding customer, retrieving a customer, retrieving all the customers, retrieve customers information based on the first name & last name and updating the customer address based on the customer id.

**Prerequisites to start customer service application,**

* Java JDK 8 (1.8)
* Maven compatible with JDK 8
* Java IDE – STS (Spring Tool Suite)
* Postman tool (for testing web service)

This is a standalone application that we can use independently with embedded tomcat, an embedded H2 database. This will be a simple web service with basic CRUD operations.

**Steps to be followed:**

1. Create spring boot application
2. Build & run the application.
3. Test the application through postman.
4. Check the documentation through swagger and postman
5. Test the application swagger-UI
6. Check the code coverage by ECL Emma junit plugin
7. **Create spring boot application:**

To create the spring boot application,

Setup a maven Spring Boot application with spring initialize or spring starter project by adding the minimum dependencies like web and test.

Add the below pom.xml as configuration to the application having dependencies like web, test, swagger, jpa, H2 and for logging as below,

**Pom.xml:**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<project xmlns=*"http://maven.apache.org/POM/4.0.0"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>com.rabo.customerservice</groupId>

<artifactId>customerservice</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>customerservice</name>

<description>customerservice Application</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.2.2.RELEASE</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger2</artifactId>

<version>2.9.2</version>

</dependency>

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger-ui</artifactId>

<version>2.7.0</version>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.apache.commons</groupId>

<artifactId>commons-lang3</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

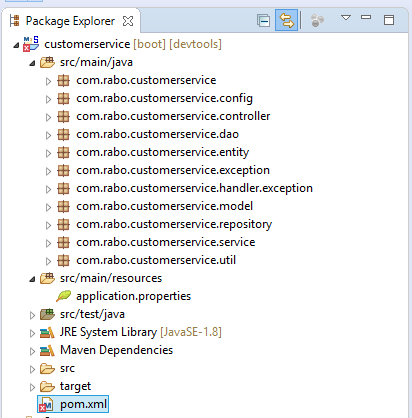
</plugin>

</plugins>

</build>

</project>

**Create the application structure:** Create the application structure as below,



**Controller package**: to controll the request and response flow

**Config package**: contains configuration like swagger.

**Dao package**: used to define the data access logic (persistence logic)

**Entity package**: used to define entity classes to interact with DB

**Exception package**: used to define user defined exceptions

**Handler package**: used to handle user defined and default exception handler classes.

**Model package**: used as data transfer among the layers.

**Repository package**: used to define the jpa repositories.

**Service package**: contains service logic between controller and Dao

**Util package**: contains utility classes

**Src/main/resources:** application.properties

Application.properties file contains the default configuration as server port, datasource configuration, connection pooling information, hibernate configuration

**Src/test/java:** contains all the Junit and Integration unit cases.

Add the source code as per the above structure.

**Now have to follow the steps build, run the spring boot application, test and code coverage and check the swagger documentation.**

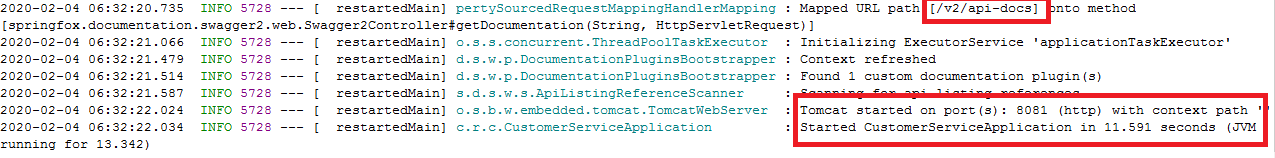
1. **Build and run:**

* **Build:**

Run the 24 Test cases and application is build successfully.



* **Run the SpringBoot Application:**

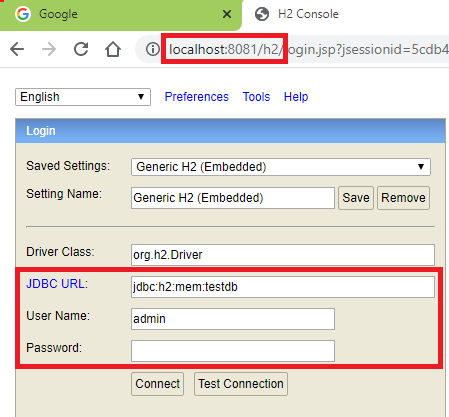


* **Login to H2 DB:**

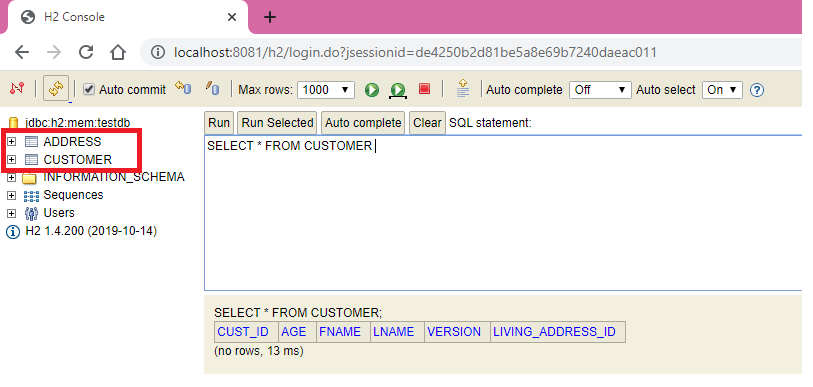
After successful build and run login to H2

http://localhost:8081/h2

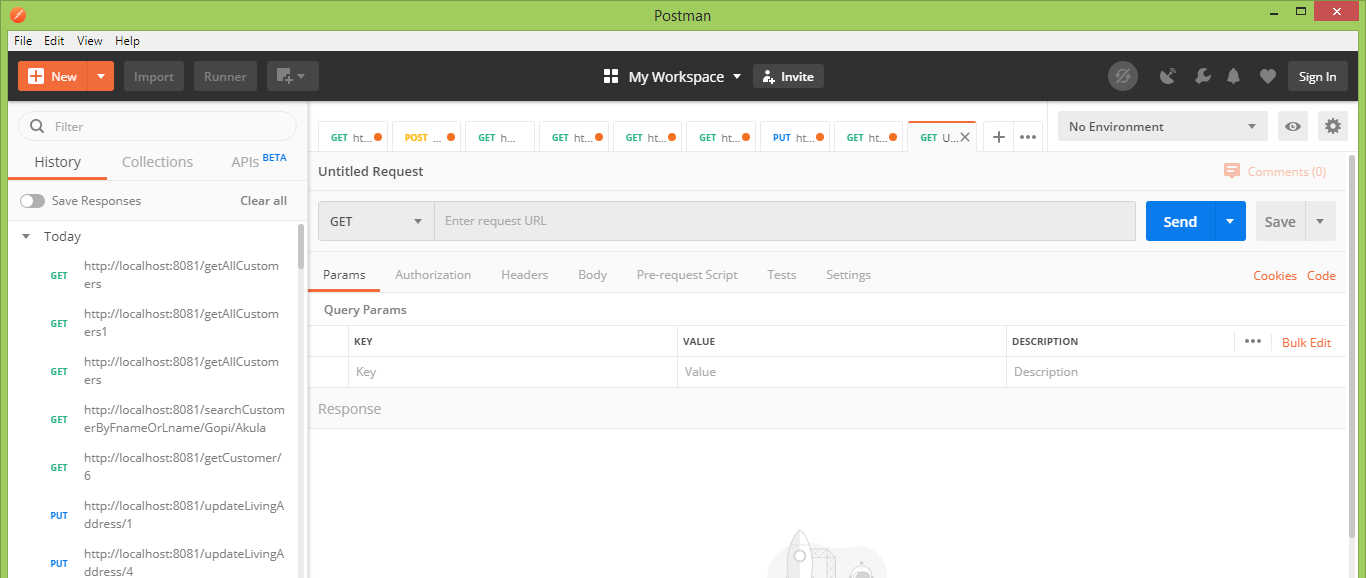
Use Password : admin



**Customer and address tables:**



**Postman :** To test the application



1. **Testing through Postman:**
2. **Functionality:** adding a new Customer

**Service url:** http://localhost:8081/customer

**Method name:** POST

**Input:** Customer info

{

"custId":2,

"fname": "A Gopi",

"lname": "AKula",

"age": 26,

"address": {

"street": "Street1",

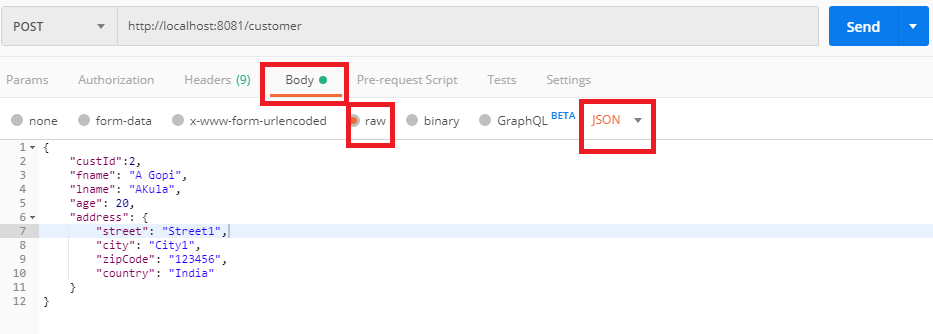
"city": "City1",

"zipCode": "123456",

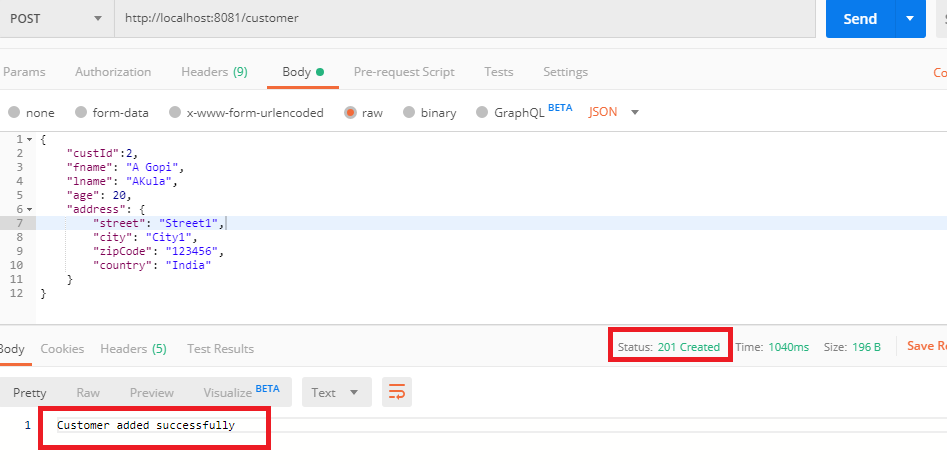
"country": "India"

}

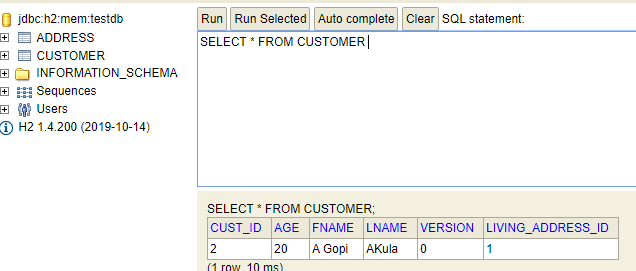
}



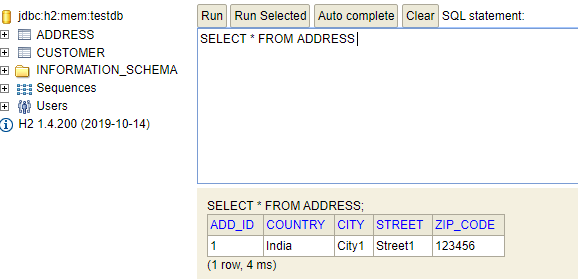
**Adding:**



**From Customer table:**



**From address Table:**



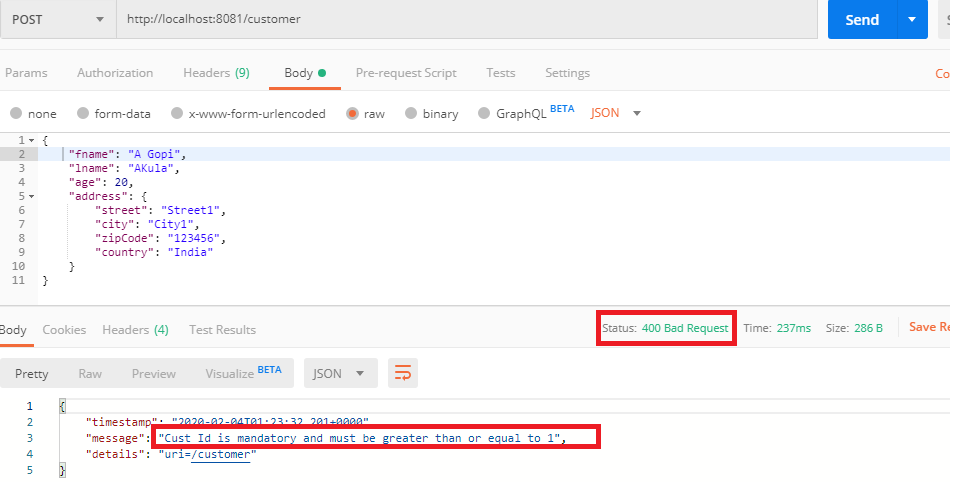
**Validations used:**

In adding customer info:

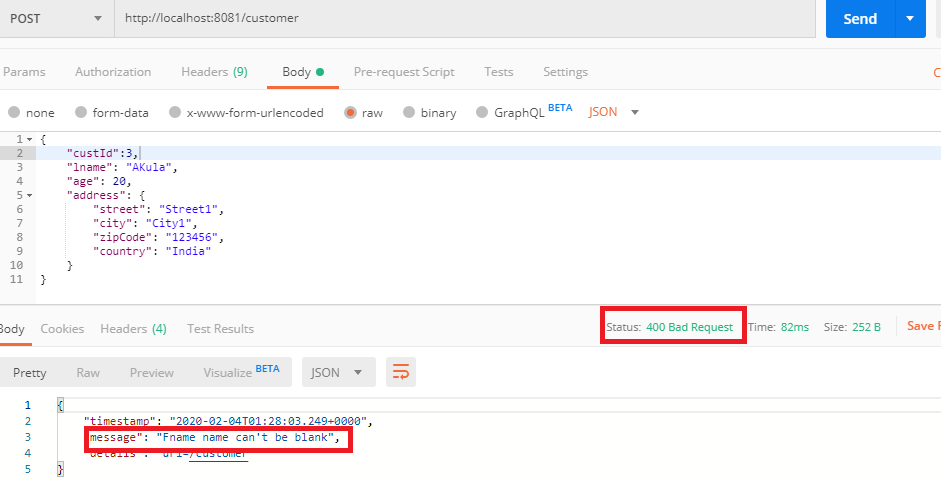
From Customer: custId, fname, lname and address are mandatory.

From Address: City, zipCode and country are mandatory.

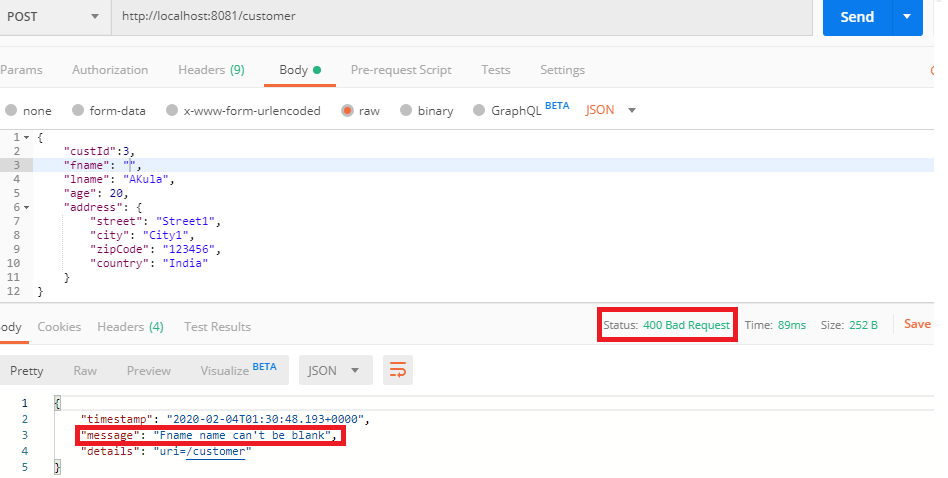
**In the absence of custid:**



**In the absence of fname:**



**In the absence of fname value:**



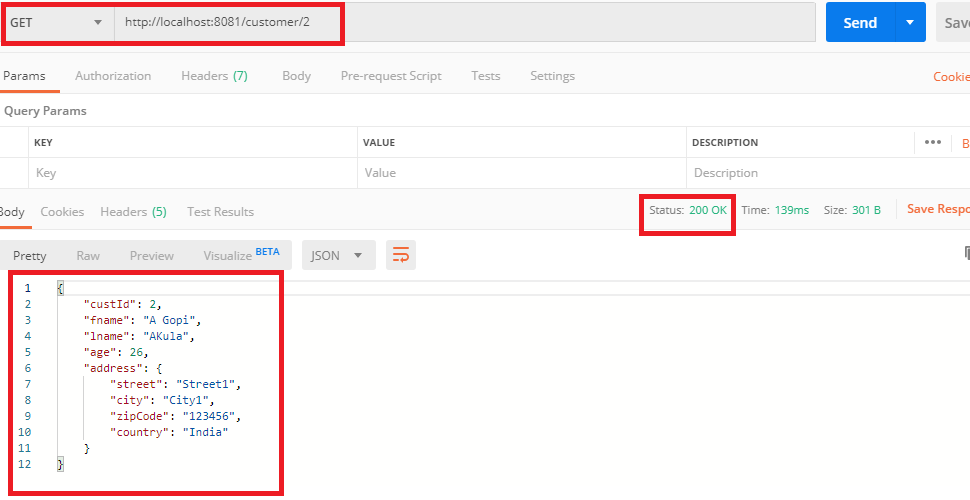
**Same for remaining validations too.**

1. **Functionality:** Retrieve a Customer based on customer id.

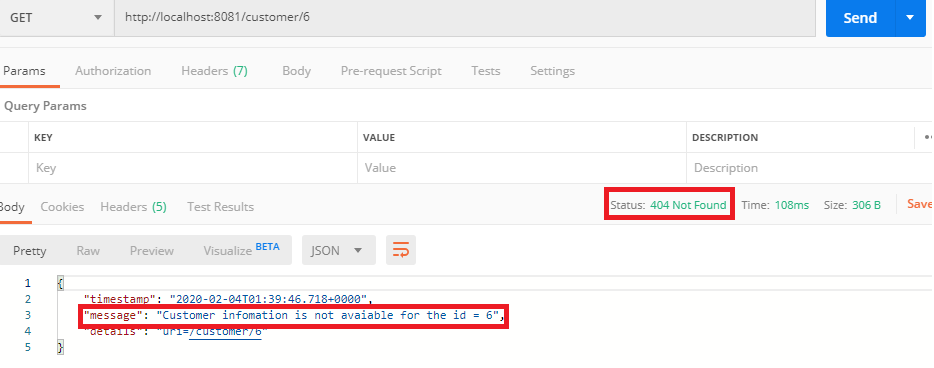
**Service url:** http://localhost:8081/customer/2

**Method name:** GET

**Input:** Customer id by url



**If customer info is not available for the given customer id:** 6



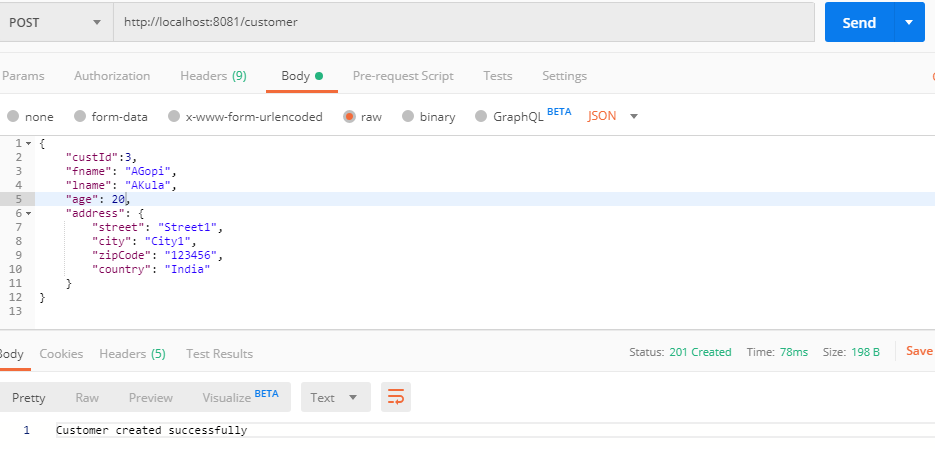
1. **Functionality:** Retrieve all the Customers

**Service url:** http://localhost:8081/customers

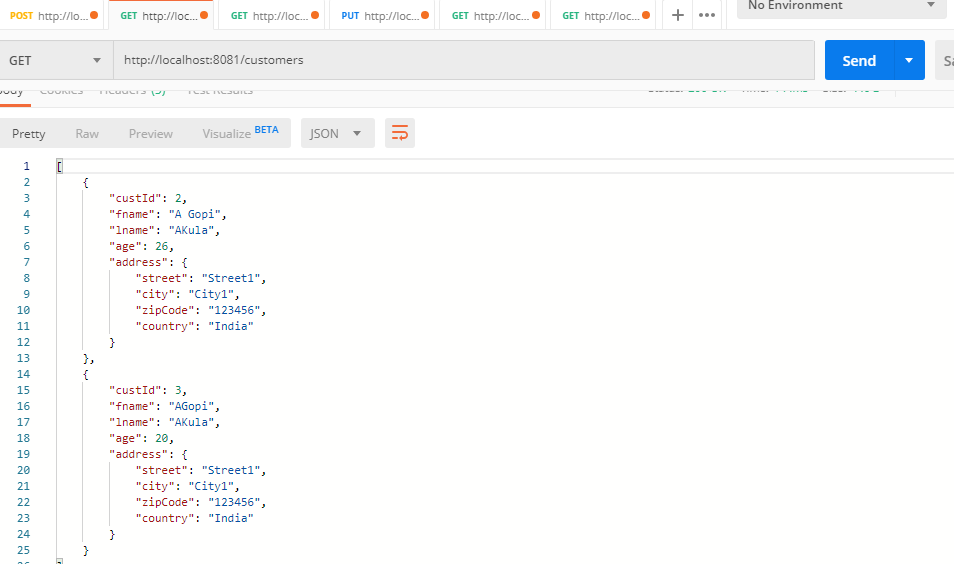
**Method name:** GET

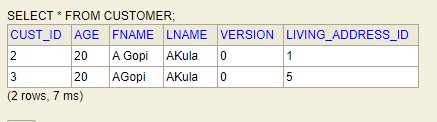
**Input:** Not required

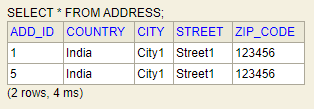
Add a new customer with customer id 3 as below,



Now check retrieve all customers,







1. **Functionality:** search by Customer name

**Service url:** http://localhost:8081/customerNames/ A Gopi/akula

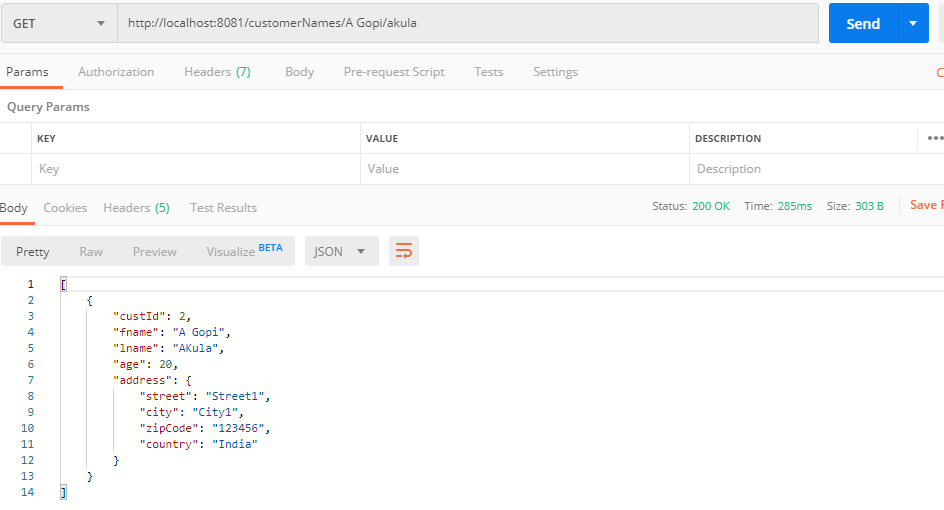
**Method name:** GET

**Input:** fname and lname by url

**From input :**

fname: A Gopi (space between A and Gopi)

lname: Akula

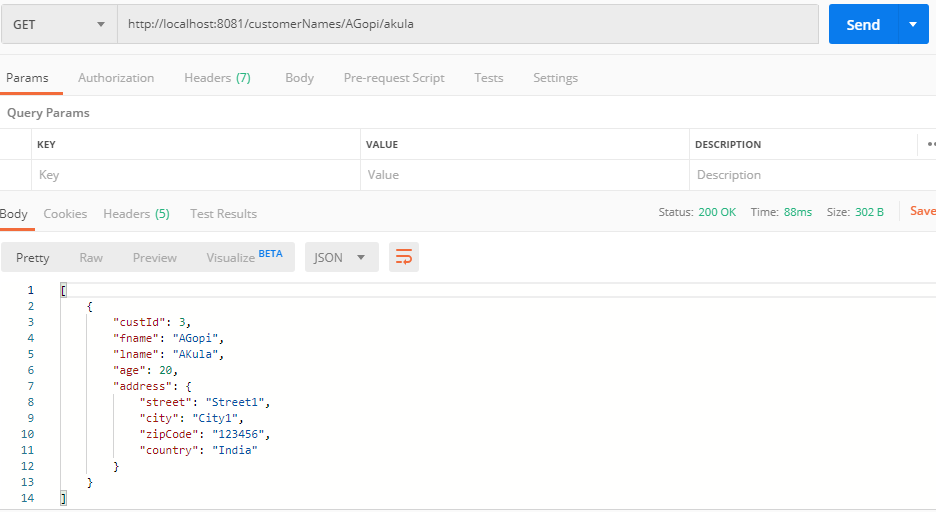


**From input :**

**fname :** AGopi (No space between A and Gopi)

**lname:** Akula

http://localhost:8081/customerNames/ AGopi/akula

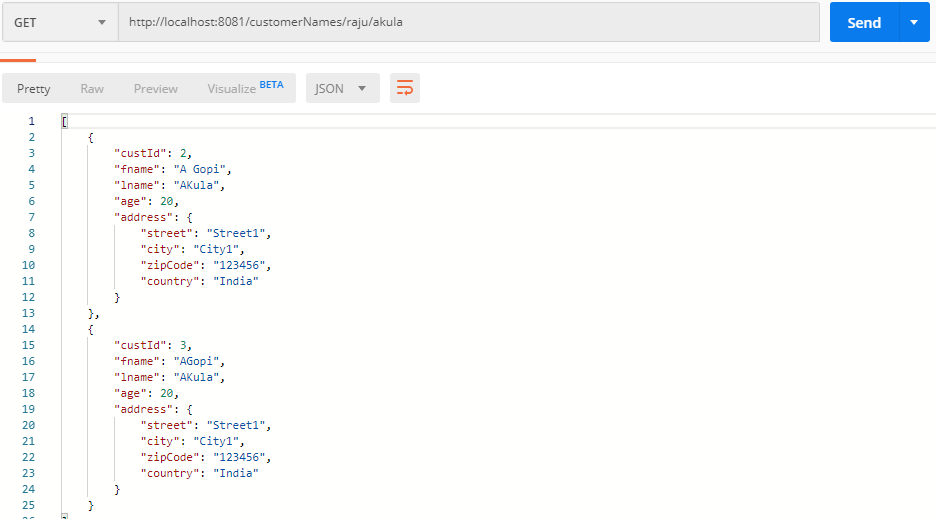


**From input** : fname or lame

**fname** : Raju ( not available in DB)

**lname:** Akula ( lname matches for 2 records)

http://localhost:8081/customerNames/ Raju/akula



1. **Functionality:** update customer address by customer id

**Service url:** http://localhost:8081/customerNames/ A Gopi/akula

**Method name:** PUT

**Input:** customer id and address json

Cust id : 2

Address: Here in updating address, address mandatory

{

"street": "Streetupdatedio",

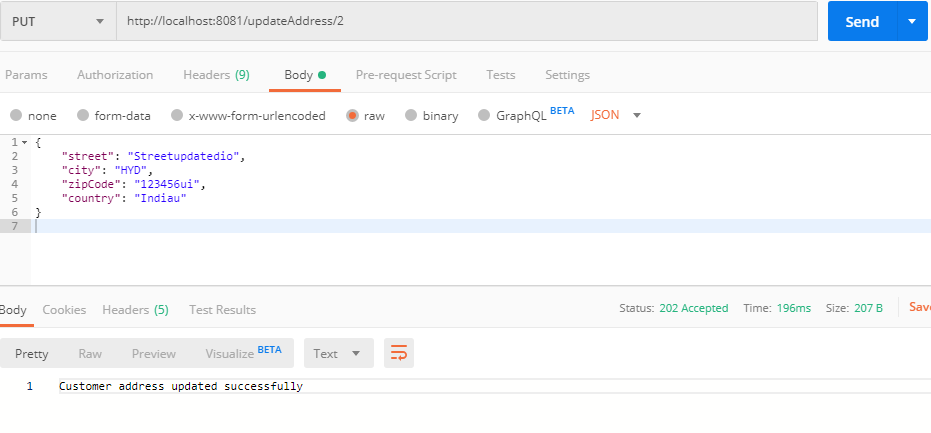
"city": "HYD",

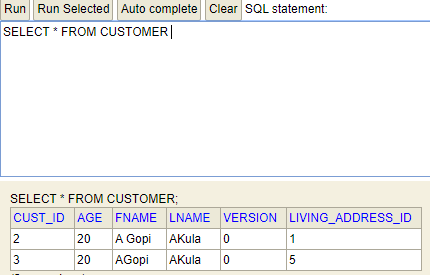
"zipCode": "123456ui",

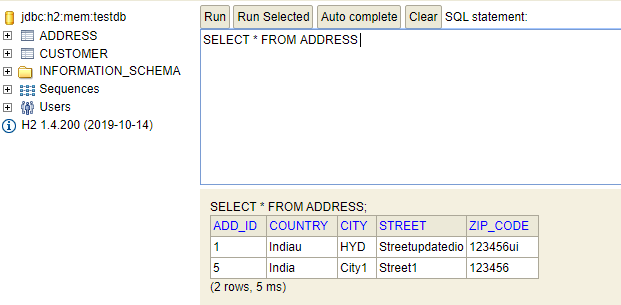
"country": "Indiau"

}

Successful l update as below,



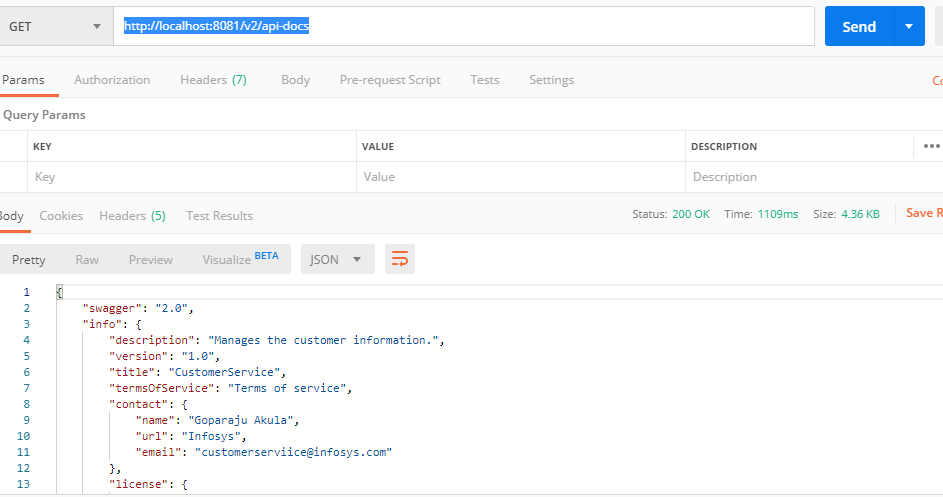




1. **Documentation generation with Swagger by postman tool:**

From post man tool use the url with get method as

<http://localhost:8081/v2/api-docs>



**Complete documentation in json format as:**

{

"swagger": "2.0",

"info": {

"description": "Manages the customer information.",

"version": "1.0",

"title": "CustomerService",

"termsOfService": "Terms of service",

"contact": {

"name": "Goparaju Akula",

"url": "Infosys",

"email": "customerserviice@infosys.com"

},

"license": {

"name": "License of API",

"url": "API license URL"

}

},

"host": "localhost:8081",

"basePath": "/",

"tags": [

{

"name": "customer-service-controller",

"description": "Customer Service Controller"

}

],

"paths": {

"/customer": {

"post": {

"tags": [

"customer-service-controller"

],

"summary": "New Customer registration",

"description": "Add the customer to DB",

"operationId": "addCustomerUsingPOST",

"consumes": [

"application/json"

],

"produces": [

"\*/\*"

],

"parameters": [

{

"in": "body",

"name": "customer",

"description": "customer",

"required": true,

"schema": {

"$ref": "#/definitions/CustomerModel"

}

}

],

"responses": {

"200": {

"description": "OK",

"schema": {

"type": "string"

}

},

"201": {

"description": "Created"

},

"401": {

"description": "Unauthorized"

},

"403": {

"description": "Forbidden"

},

"404": {

"description": "Not Found"

}

},

"deprecated": false

}

},

"/customer/{id}": {

"get": {

"tags": [

"customer-service-controller"

],

"summary": "Retrieves single customer information",

"description": "Based on given customer id, customer information is retrieved",

"operationId": "getCustomerUsingGET",

"produces": [

"application/json"

],

"parameters": [

{

"name": "id",

"in": "path",

"description": "id",

"required": true,

"type": "integer",

"format": "int32"

}

],

"responses": {

"200": {

"description": "OK",

"schema": {

"$ref": "#/definitions/WildcardType"

}

},

"401": {

"description": "Unauthorized"

},

"403": {

"description": "Forbidden"

},

"404": {

"description": "Not Found"

}

},

"deprecated": false

}

},

"/customerNames/{fname}/{lname}": {

"get": {

"tags": [

"customer-service-controller"

],

"summary": "Retrieves customer information by fname and/or lname",

"description": "Based on given customer fname and/or lanme, customer information is retrieved",

"operationId": "searchCustomerNameByUsingGET",

"produces": [

"application/json"

],

"parameters": [

{

"name": "fname",

"in": "path",

"description": "fname",

"required": true,

"type": "string"

},

{

"name": "lname",

"in": "path",

"description": "lname",

"required": true,

"type": "string"

}

],

"responses": {

"200": {

"description": "OK",

"schema": {

"type": "array",

"items": {

"type": "object"

}

}

},

"401": {

"description": "Unauthorized"

},

"403": {

"description": "Forbidden"

},

"404": {

"description": "Not Found"

}

},

"deprecated": false

}

},

"/customers": {

"get": {

"tags": [

"customer-service-controller"

],

"summary": "Retrieves all the customers information",

"description": "All the customers information available in is retrieved",

"operationId": "getAllCustomersUsingGET",

"produces": [

"application/json"

],

"responses": {

"200": {

"description": "OK",

"schema": {

"type": "array",

"items": {

"type": "object"

}

}

},

"401": {

"description": "Unauthorized"

},

"403": {

"description": "Forbidden"

},

"404": {

"description": "Not Found"

}

},

"deprecated": false

}

},

"/updateAddress/{id}": {

"put": {

"tags": [

"customer-service-controller"

],

"summary": "Updates the customer living address",

"description": "Based on given customer id and address, address is updated for the given customer id",

"operationId": "updateLivingAddressUsingPUT",

"consumes": [

"application/json"

],

"produces": [

"\*/\*"

],

"parameters": [

{

"in": "body",

"name": "address",

"description": "address",

"required": true,

"schema": {

"$ref": "#/definitions/AddressModel"

}

},

{

"name": "id",

"in": "path",

"description": "id",

"required": true,

"type": "integer",

"format": "int32"

}

],

"responses": {

"200": {

"description": "OK",

"schema": {

"type": "string"

}

},

"201": {

"description": "Created"

},

"401": {

"description": "Unauthorized"

},

"403": {

"description": "Forbidden"

},

"404": {

"description": "Not Found"

}

},

"deprecated": false

}

}

},

"definitions": {

"AddressModel": {

"type": "object",

"properties": {

"city": {

"type": "string"

},

"country": {

"type": "string"

},

"id": {

"type": "integer",

"format": "int32"

},

"street": {

"type": "string"

},

"zipCode": {

"type": "string"

}

},

"title": "AddressModel"

},

"CustomerModel": {

"type": "object",

"properties": {

"address": {

"$ref": "#/definitions/AddressModel"

},

"age": {

"type": "integer",

"format": "int32"

},

"custId": {

"type": "integer",

"format": "int32"

},

"fname": {

"type": "string"

},

"lname": {

"type": "string"

}

},

"title": "CustomerModel"

},

"Optional«CustomerModel»": {

"type": "object",

"properties": {

"present": {

"type": "boolean"

}

},

"title": "Optional«CustomerModel»"

},

"WildcardType": {

"type": "object",

"properties": {

"present": {

"type": "boolean"

}

},

"title": "WildcardType"

}

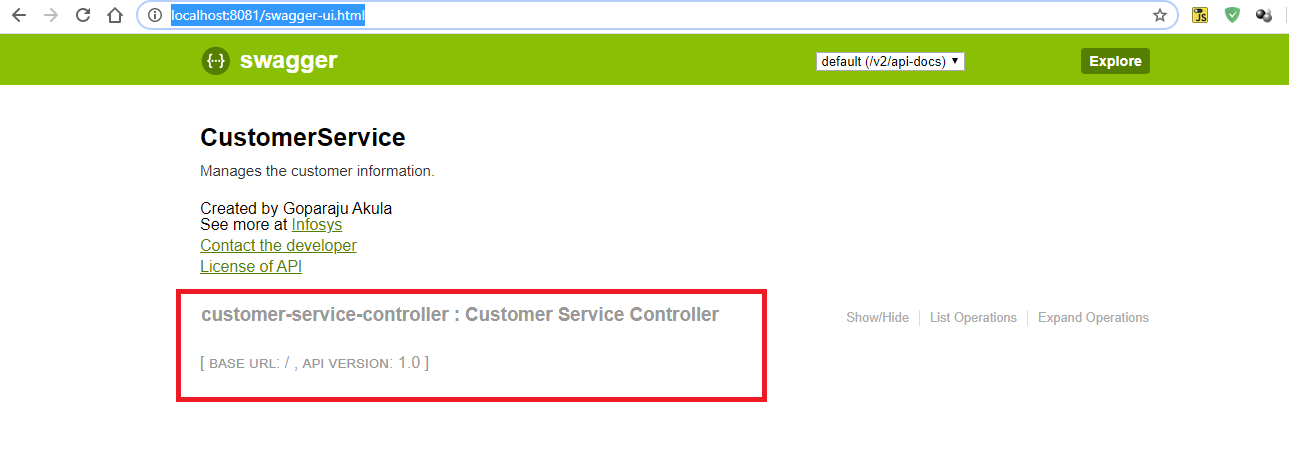
}

}

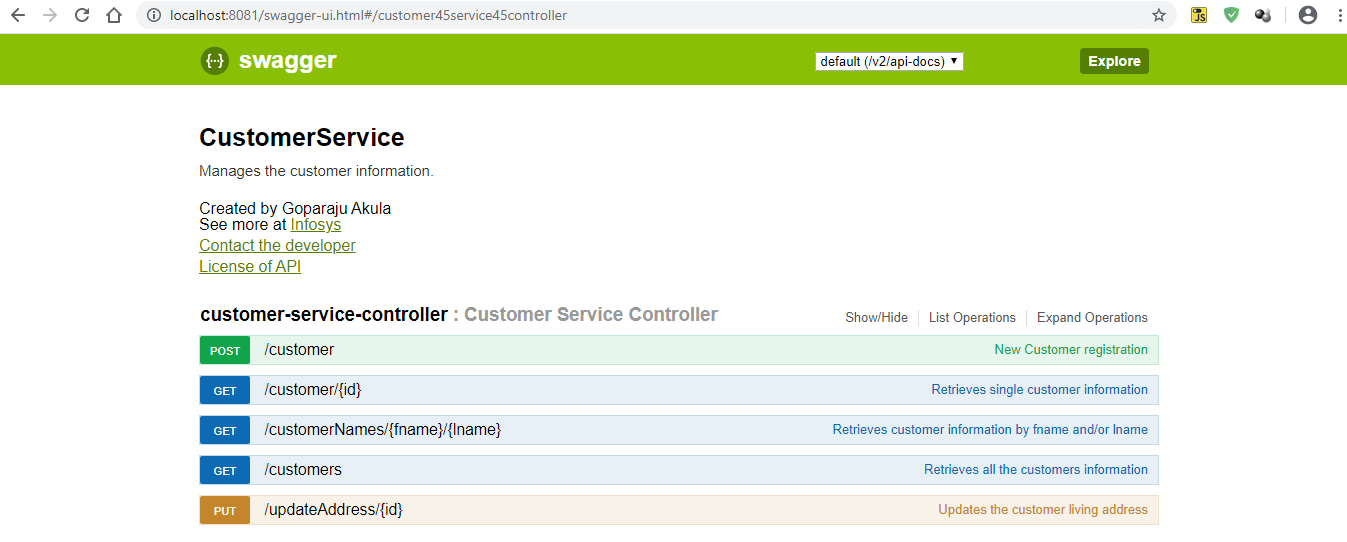
1. **Documentation generation with Swagger by swagger - UI:**

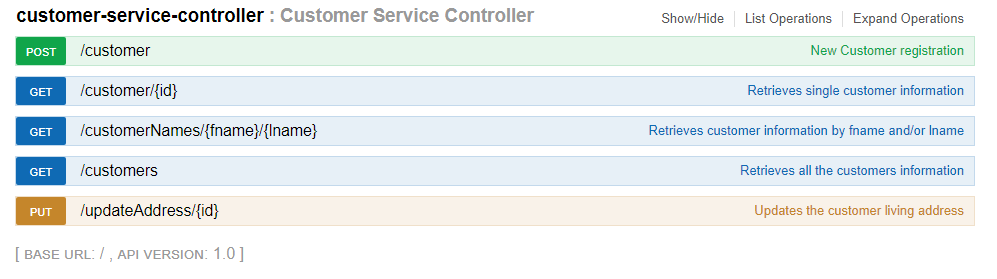
From browser: <http://localhost:8081/swagger-ui.html>

Swagger for customer service as below,



Swagger UI with all rest end points as below,

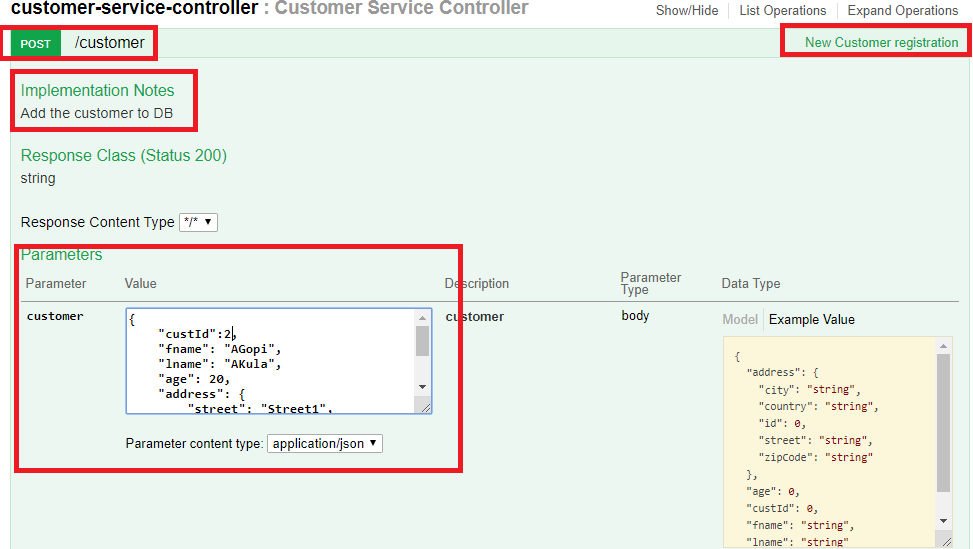


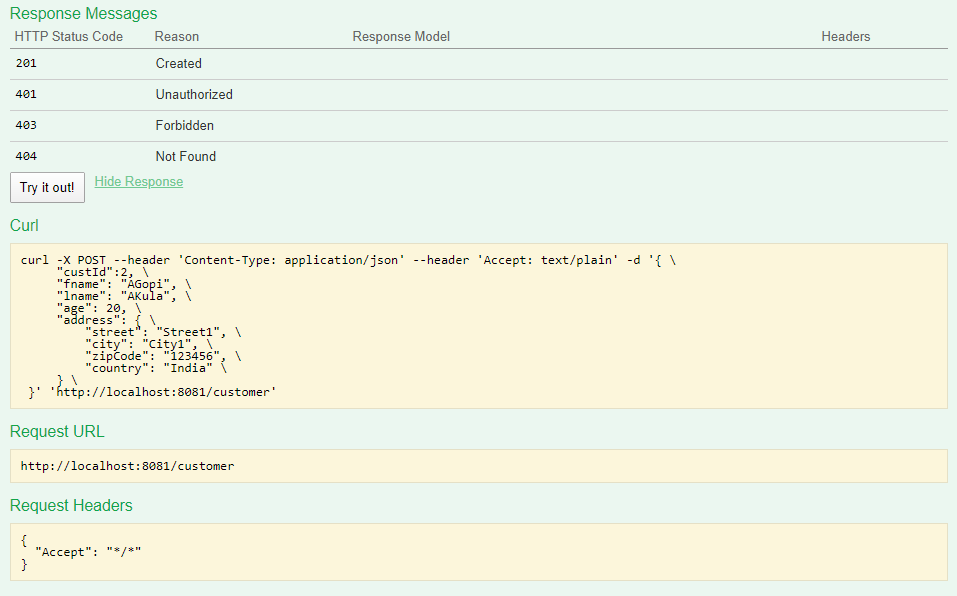


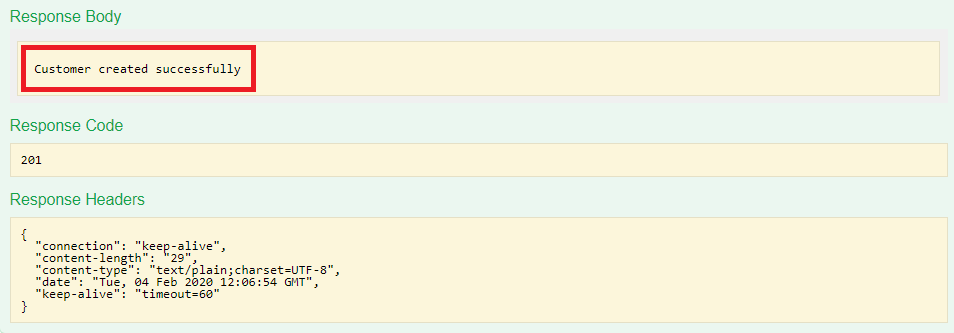
**Testing through swagger-UI:**

Testing adding a customer:

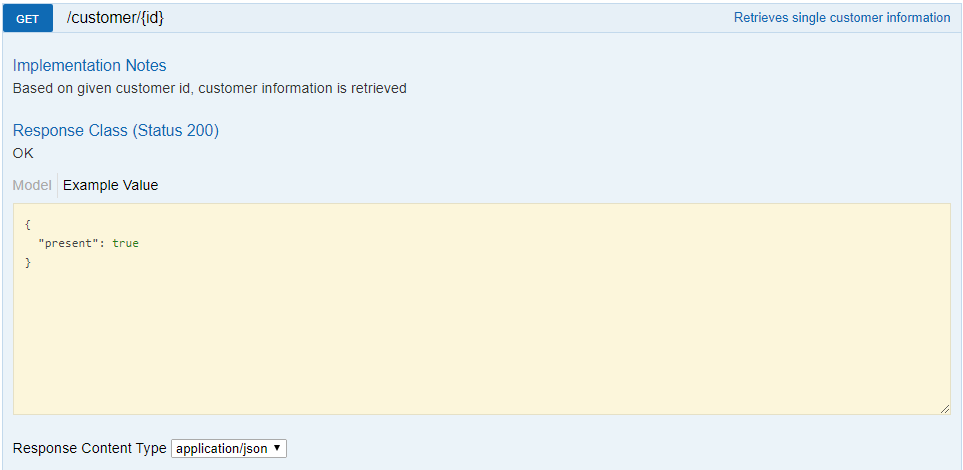
Place the input in parameters part as highlighted.

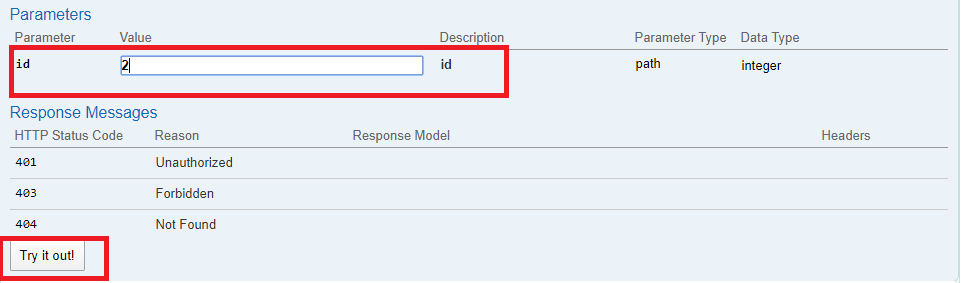


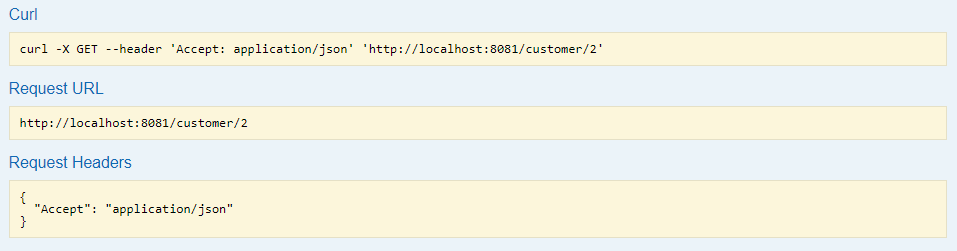


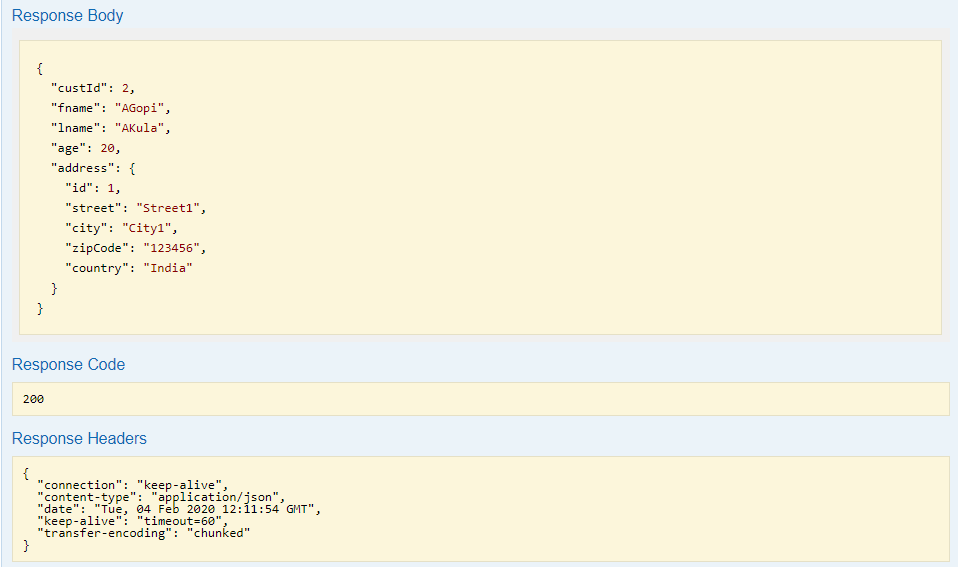


**To retrieve a customer:**

****

****

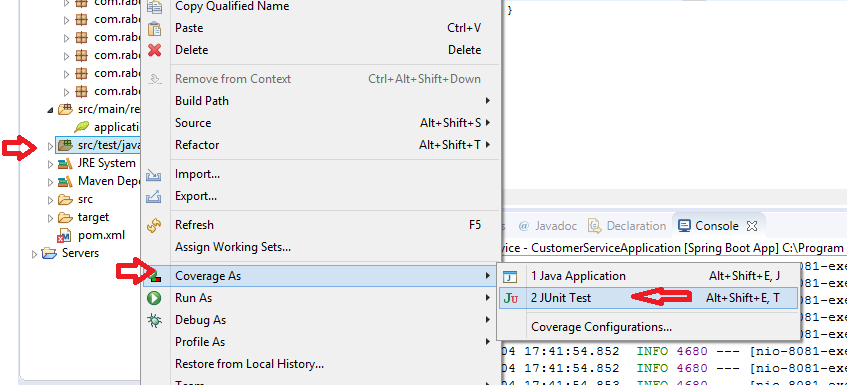
****

****

**Same procedure for remaining service also same.**

1. **Check the code coverage by ecl emma junit plugin:**

Run the code coverage on **src/mian/test** as below,



**Code coverage with 82.3%**

