Spring Boot and MongoDB

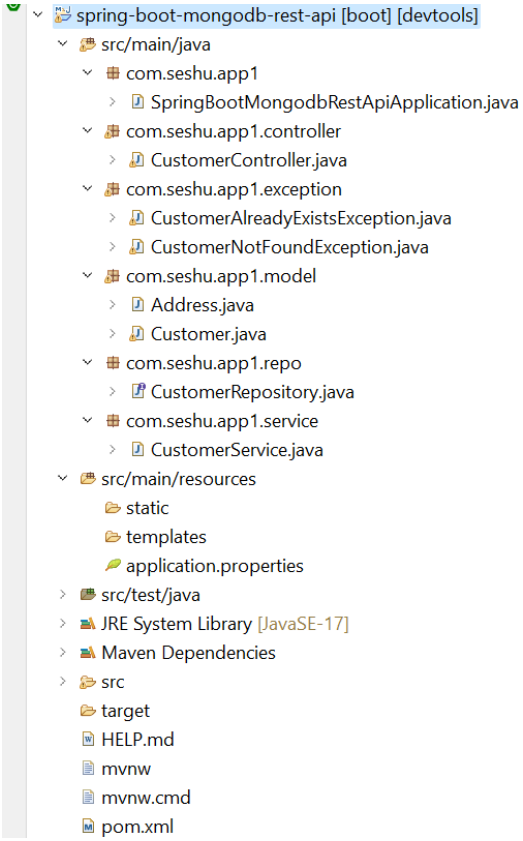
- MongoDB is the most popular NoSQL database because of the ease with which data can be stored and retrieved.

- Combining Spring Boot and MongoDB results in applications that are fast, secure, reliable, and require minimum development time.

MongoRepository:
MongoRepository is used for basic queries that involve all or many fields of the document.

Examples;
Data creation, viewing documents, and more

Example:



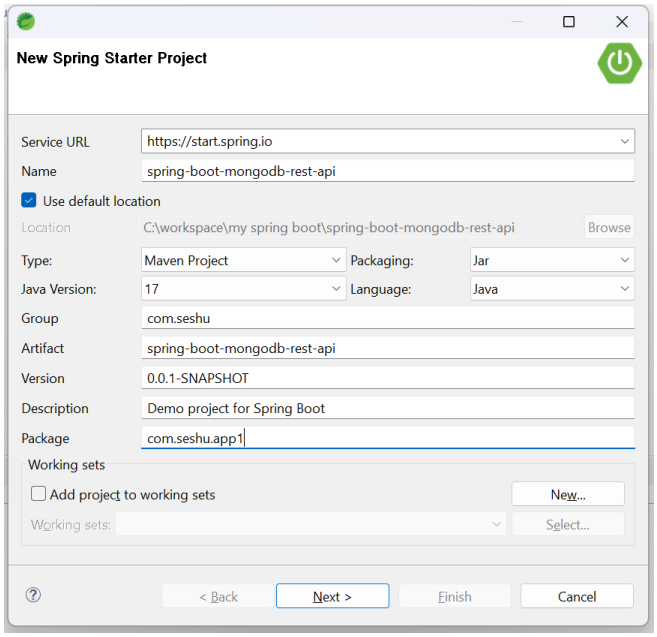
Create a new spring boot project ***spring-boot-mongodb-rest-api*** with fallowing dependencies

Spring Boot DevTools

Spring Data MongoDB

Spring Web

Lombok



src\main\resources\application.properties

```
server.port=8181
spring.data.mongodb.database=customerdb
spring.data.mongodb.uri=mongodb://localhost:27017/
server.error.include-message=always
```

Address.java

```
package com.seshu.app1.model;

import org.springframework.data.mongodb.core.mapping.Document;

import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;

@Data
@Document
@NoArgsConstructor
@RequiredArgsConstructor
public class Address {
	@NonNull
	private String city;

	@NonNull
	private String state;

	@NonNull
	private String country;
}
```

Customer.java

```
package com.seshu.app1.model;

import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;

import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;

@Data
@Document
@NoArgsConstructor
@RequiredArgsConstructor
public class Customer {
	@Id
	@NonNull
	private int customerId;
```

```
        @NonNull
        private String customerName;

        @NonNull
        private String customerEmail;

        @NonNull
        private Address customerAddress;
}
```

CustomerRepository.java

```
package com.seshu.app1.repo;

import java.util.List;

import org.springframework.data.mongodb.repository.MongoRepository;
import org.springframework.data.mongodb.repository.Query;

import com.seshu.app1.model.Customer;

public interface CustomerRepository extends MongoRepository<Customer,Integer> {
   @Query("{'customerAddress.city' : {$in : [?0]}}")
   List<Customer> findAllCustomerFromCity(String city);
}
```

CustomerAlreadyExistsException.java

```
package com.seshu.app1.exception;

import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;

@ResponseStatus(code = HttpStatus.CONFLICT , reason = "Customer already exists")
public class CustomerAlreadyExistsException extends Exception{

}
```

CustomerNotFoundException.java

```
package com.seshu.app1.exception;

import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;

@ResponseStatus(code= HttpStatus.NOT_FOUND , reason = "Customer with specified id is not found")
public class CustomerNotFoundException extends Exception{

}
```

CustomerService.java

```
package com.seshu.app1.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.seshu.app1.exception.CustomerAlreadyExistsException;
import com.seshu.app1.exception.CustomerNotFoundException;
import com.seshu.app1.model.Customer;
import com.seshu.app1.repo.CustomerRepository;

@Service
public class CustomerService {
	@Autowired
	private CustomerRepository customerRepository;

	public Customer saveCustomerDetails(Customer customer) throws CustomerAlreadyExistsException
{
		if (customerRepository.findById(customer.getCustomerId()).isPresent()) {
			throw new CustomerAlreadyExistsException();
		}
		return customerRepository.save(customer);
	}

	public boolean deleteCustomer(int customerCode) throws CustomerNotFoundException {
		boolean flag = false;
		if (customerRepository.findById(customerCode).isEmpty()) {
			throw new CustomerNotFoundException();
```

```
		} else {
			customerRepository.deleteById(customerCode);
			flag = true;
		}
		return flag;
	}

	public List<Customer> getAllCustomerDetails() {
		return customerRepository.findAll();
	}

	public List<Customer> getAllCustomersByCity(String city) {
		return customerRepository.findAllCustomerFromCity(city);
	}

	public Customer updateCustomerDetails(Customer customer) throws CustomerNotFoundException
{
		if (!customerRepository.findById(customer.getCustomerId()).isPresent()) {
			throw new CustomerNotFoundException();
		}
		return customerRepository.save(customer);
	}

}
```

CustomerController.java

```
package com.seshu.app1.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.seshu.app1.exception.CustomerAlreadyExistsException;
import com.seshu.app1.exception.CustomerNotFoundException;
import com.seshu.app1.model.Customer;
```

```
import com.seshu.app1.service.CustomerService;

@RestController
@RequestMapping("api/v1/customerservice/")
public class CustomerController {

	@Autowired
	private CustomerService customerService;

	private ResponseEntity<?> responseEntity;

	@PostMapping("customer")
	public ResponseEntity<?> saveCustomer(@RequestBody Customer customer) throws
CustomerAlreadyExistsException {
		try {
			customerService.saveCustomerDetails(customer);
			responseEntity = new ResponseEntity<>(customer, HttpStatus.CREATED);
		} catch (CustomerAlreadyExistsException e) {
			throw new CustomerAlreadyExistsException();
		} catch (Exception e) {
			responseEntity = new ResponseEntity<>("Error  !!!Try after sometime",
HttpStatus.INTERNAL_SERVER_ERROR);
		}
		return responseEntity;
	}

	@GetMapping("customer")
	public ResponseEntity<?> getAllCustomer() {
		try {
			responseEntity = new ResponseEntity<>(customerService.getAllCustomerDetails(),
HttpStatus.OK);
		} catch (Exception e) {
			responseEntity = new ResponseEntity<>("Error !!! Try after sometime.",
HttpStatus.INTERNAL_SERVER_ERROR);
		}
		return responseEntity;
	}

	@GetMapping("customer/{city}")
	public ResponseEntity<?> getAllCustomerByCity(@PathVariable String city) {
		try {
			responseEntity = new
ResponseEntity<>(customerService.getAllCustomersByCity(city), HttpStatus.OK);
		} catch (Exception e) {
```

```
                        responseEntity = new ResponseEntity<>("Error !!! Try after sometime.",
HttpStatus.INTERNAL_SERVER_ERROR);
                }
                return responseEntity;
        }

        @DeleteMapping("customer/{customerId}")
        public ResponseEntity<?> deleteCustomer(@PathVariable("customerId") int customerId)
                        throws CustomerNotFoundException {

                try {
                        customerService.deleteCustomer(customerId);
                        responseEntity = new ResponseEntity<>("Successfully deleted !!!", HttpStatus.OK);
                } catch (CustomerNotFoundException e) {
                        throw new CustomerNotFoundException();
                } catch (Exception exception) {
                        responseEntity = new ResponseEntity<>("Error !!! Try after sometime.",
HttpStatus.INTERNAL_SERVER_ERROR);
                }
                return responseEntity;
        }

        @PutMapping("customer")
        public ResponseEntity<?> updateCustomer(@RequestBody Customer customer) throws
CustomerNotFoundException {
                try {
                        customerService.updateCustomerDetails(customer);
                        responseEntity = new ResponseEntity<>(customer, HttpStatus.CREATED);
                } catch (CustomerNotFoundException e) {
                        throw new CustomerNotFoundException();
                } catch (Exception e) {
                        responseEntity = new ResponseEntity<>("Error  !!!Try after sometime",
HttpStatus.INTERNAL_SERVER_ERROR);
                }
                return responseEntity;
        }
}
```

CustomerServiceApplication.java

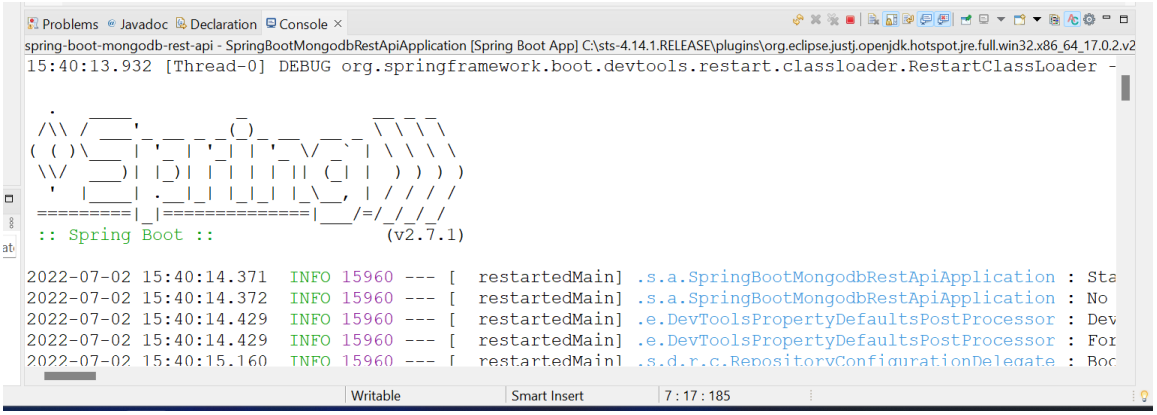
```
package com.seshu.app1;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class SpringBootMongodbRestApiApplication {

	public static void main(String[] args) {
		SpringApplication.run(SpringBootMongodbRestApiApplication.class, args);
	}
}
```

Run the starter class.



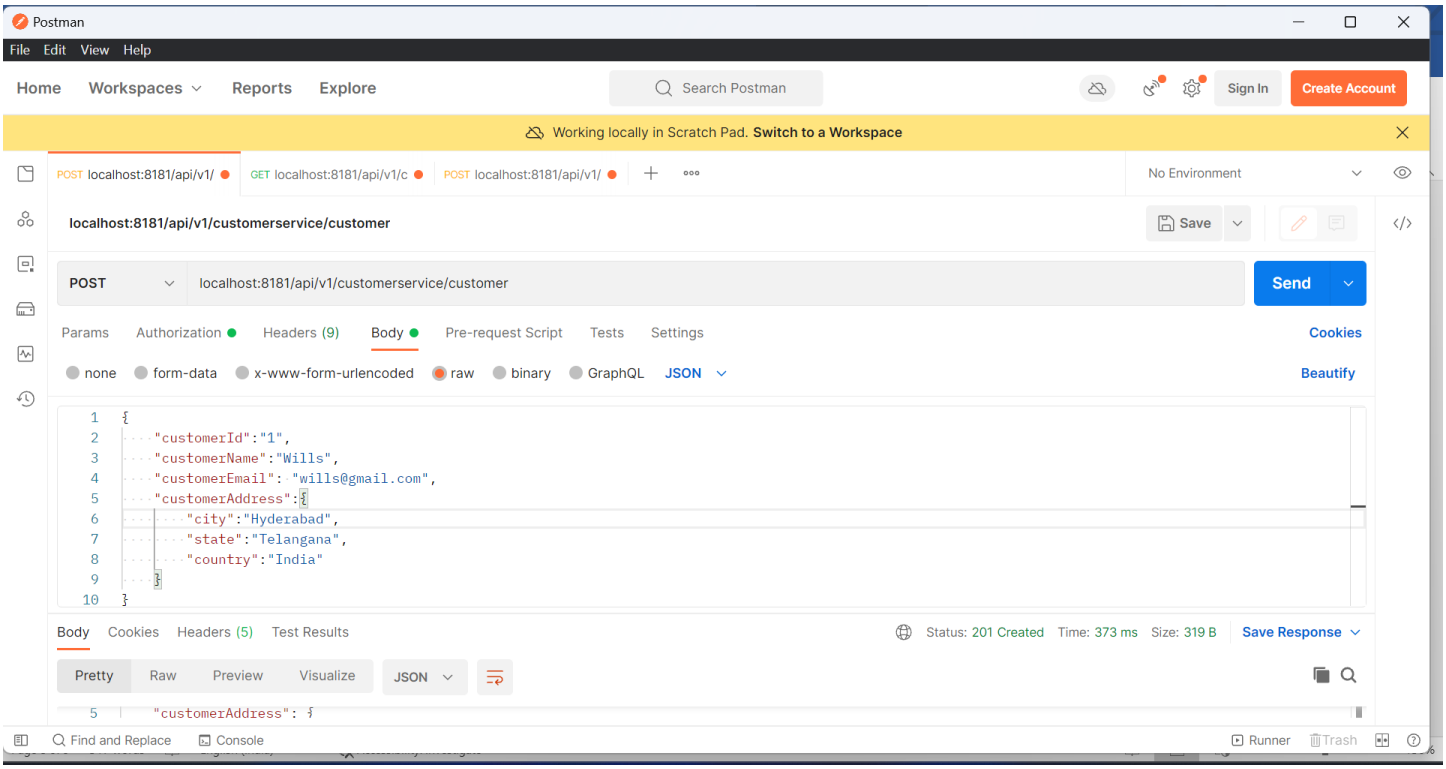
Test using Postman

Post Request

localhost:8181/api/v1/customerservice/customer

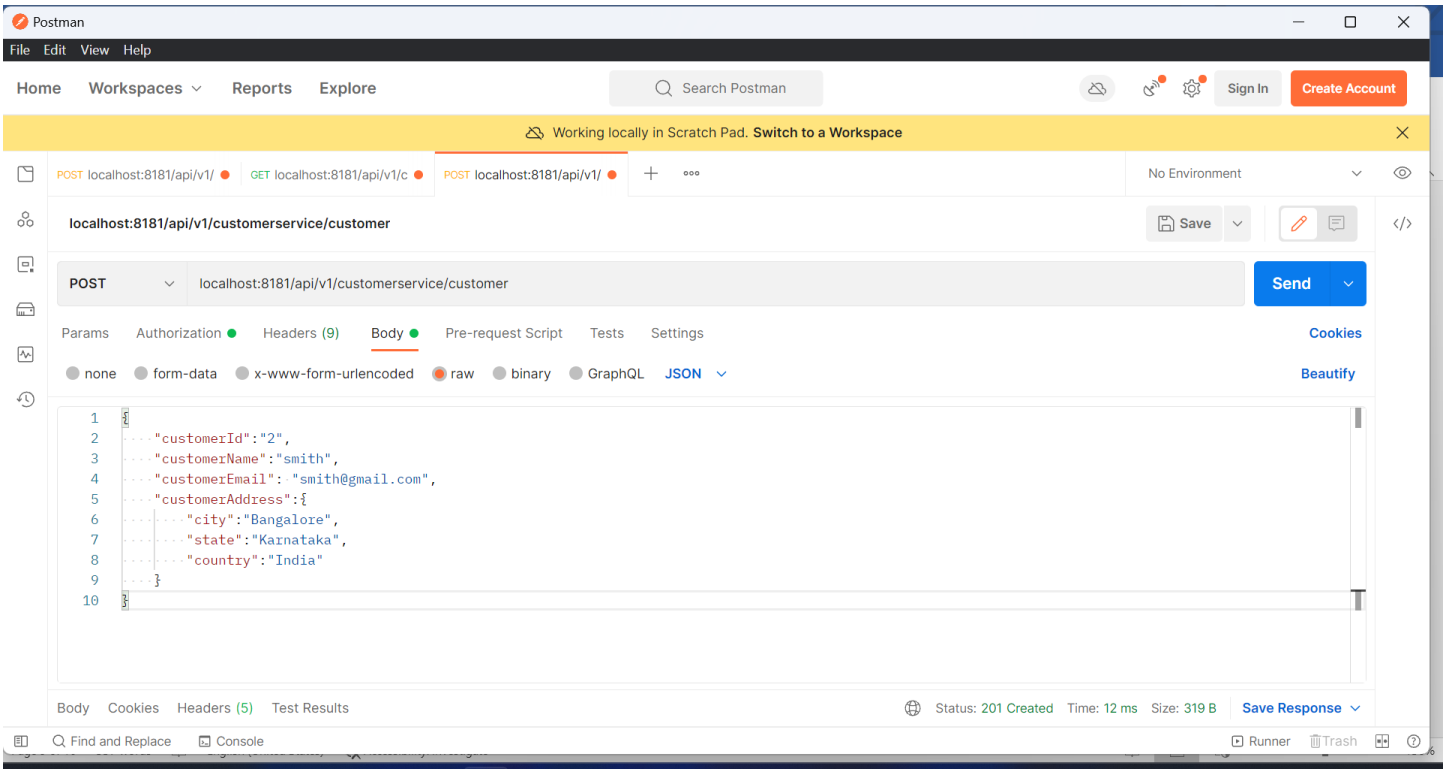
Adding Customer 1 details (add data in Body)

```
{
  "customerId":"1",
  "customerName":"Wills",
  "customerEmail": "wills@gmail.com",
  "customerAddress":{
    "city":"Hyderabad",
    "state":"Telangana",
    "country":"India"
  }
}
```



Adding customer 2 details (add data in Body)

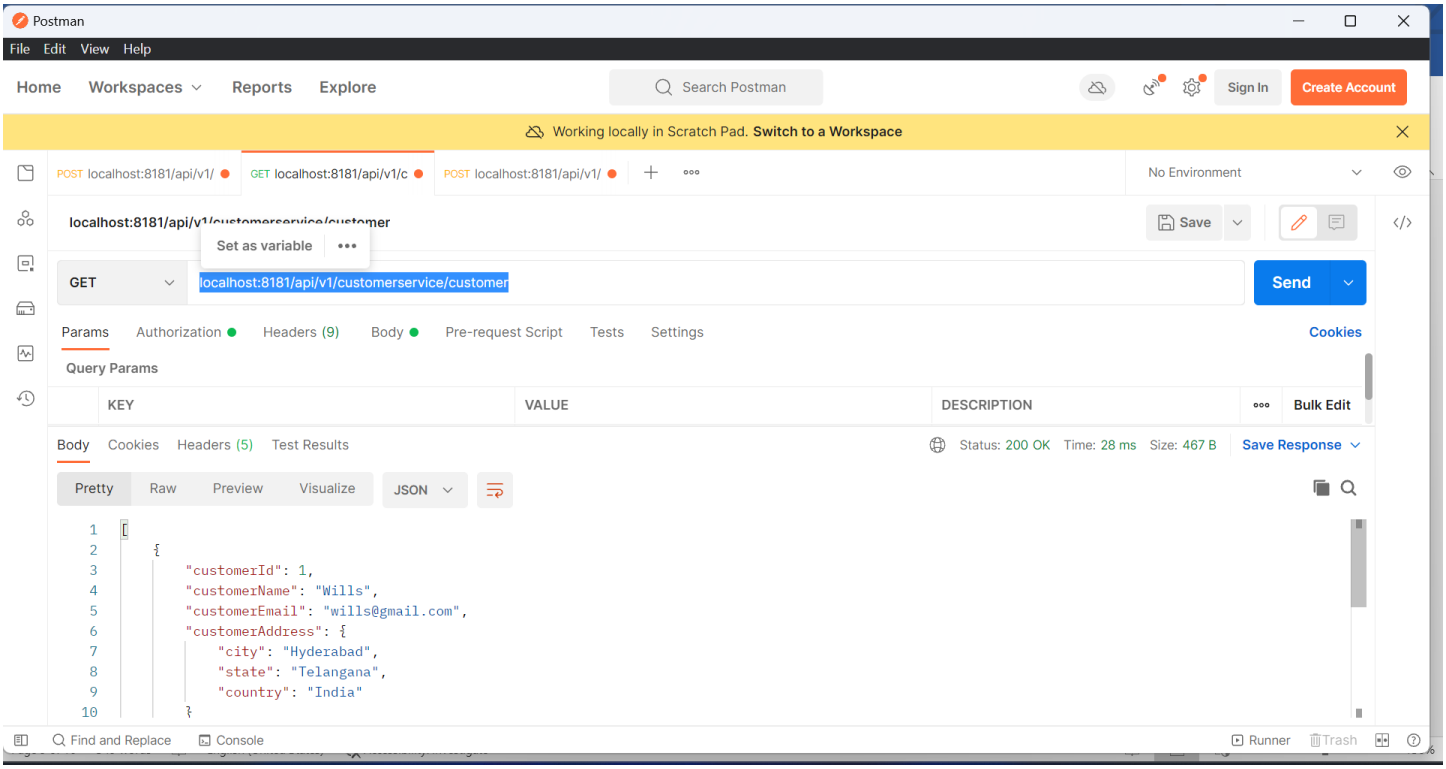
```
{
   "customerId":"2",
   "customerName":"smith",
   "customerEmail": "smith@gmail.com",
   "customerAddress":{
      "city":"Bangalore",
      "state":"Karnataka",
      "country":"India"
   }
}
```



Handling Get Request;

(Get all customer details)

localhost:8181/api/v1/customerservice/customer



Check in MongoDB Server;

Execute these commands in CMD

```
>mongo
> use customerdb
switched to db customerdb

> show collections
customer

> db.customer.find().pretty()
```

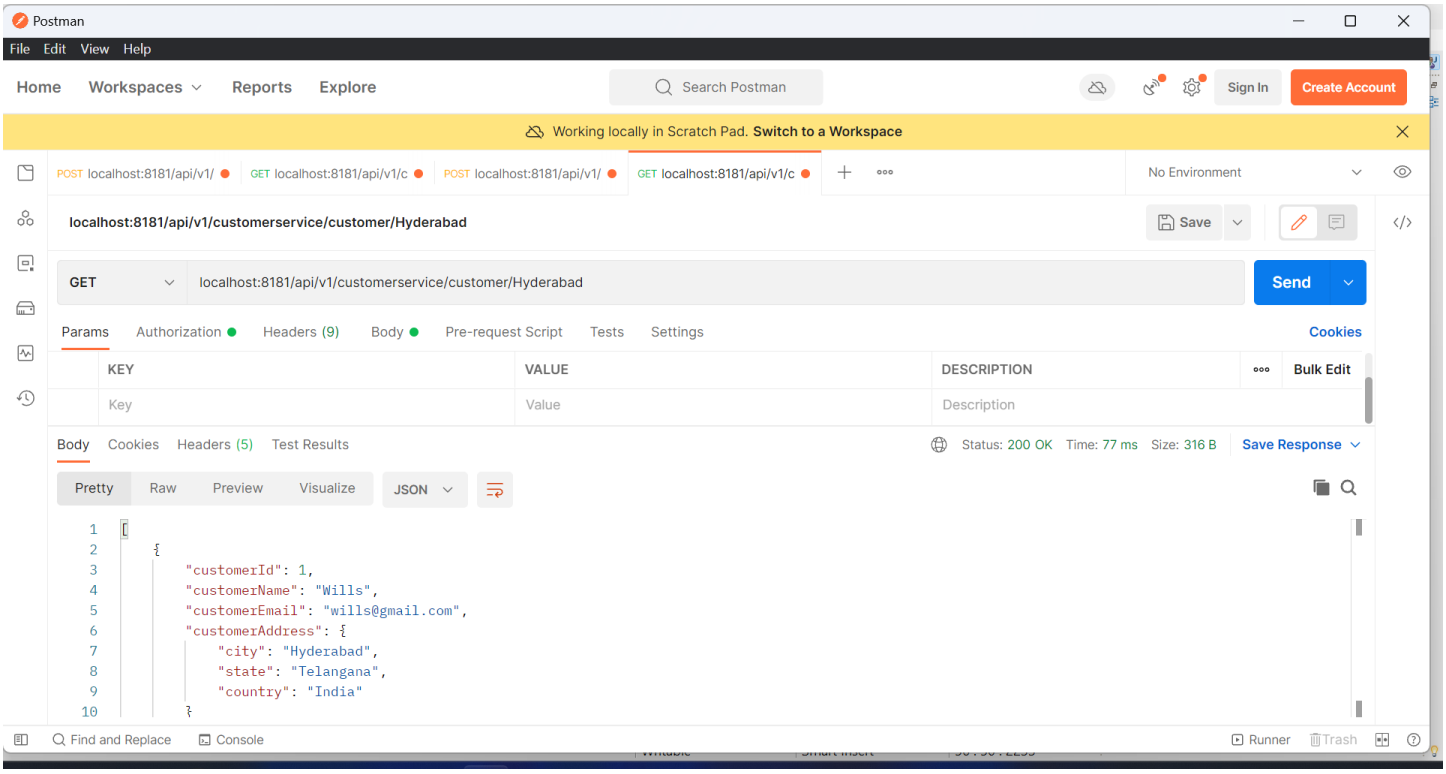
Note: To drop existing database

```
> use customerdb
> db.dropDatabase()
```

Handling Get Request;

(Get Customer details based on city)

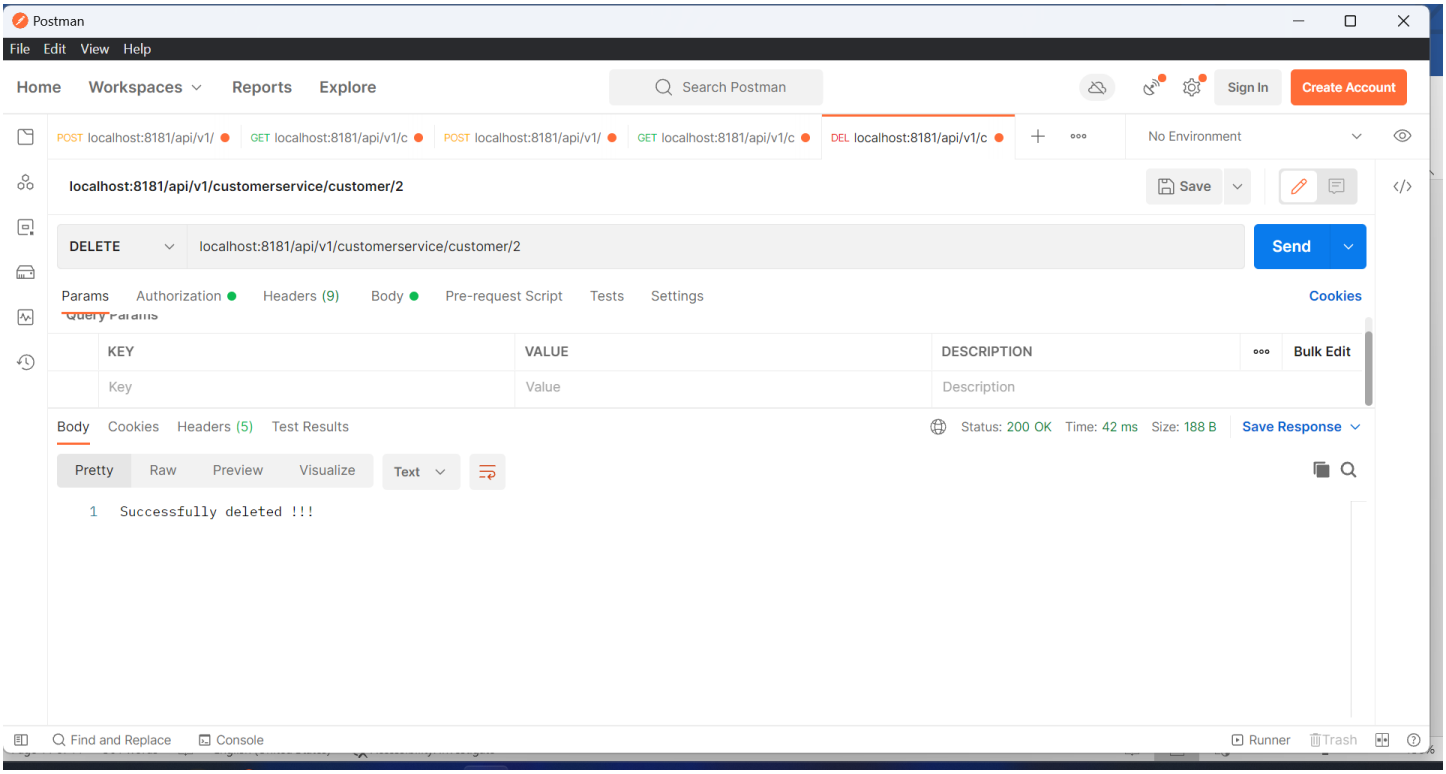
http://localhost:8181/api/v1/customerservice/customer/Hyderabad



Handling Delete Request;

(Delete customer by customer id)

localhost:8181/api/v1/customerservice/customer/2

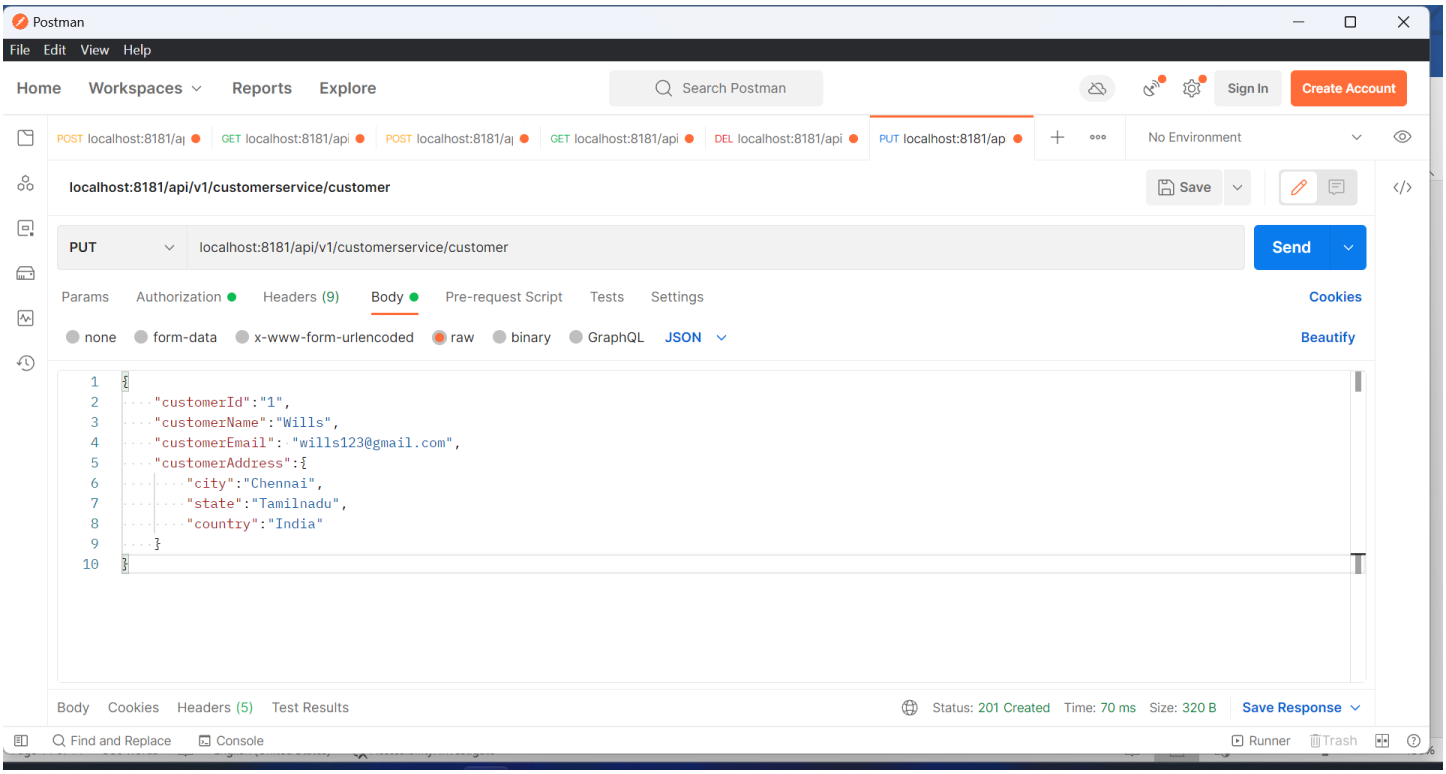


Handling Put Request

(Update customer details)

localhost:8181/api/v1/customerservice/customer

```
{
   "customerId":"1",
   "customerName":"Wills",
   "customerEmail": "wills123@gmail.com",
   "customerAddress":{
      "city":"Chennai",
      "state":"Tamilnadu",
      "country":"India"
   }
}
```



Updated information;