[Thực hành] Quản lý khách hàng sử dụng RESTful

Mục tiêu

Luyện tập triển khai RESTful webservice trong một ứng dụng Spring MVC.

Mô tả

Trong phần này, chúng ta sẽ phát triển một ứng dụng quản lý khách hàng, ứng dụng này chỉ bao gồm phần back-end, cung cấp các API để thao tác với khách hàng:

* Trả về danh sách khách hàng
* Thêm khách hàng mới
* Xoá khách hàng
* Chỉnh sửa thông tin khách hàng

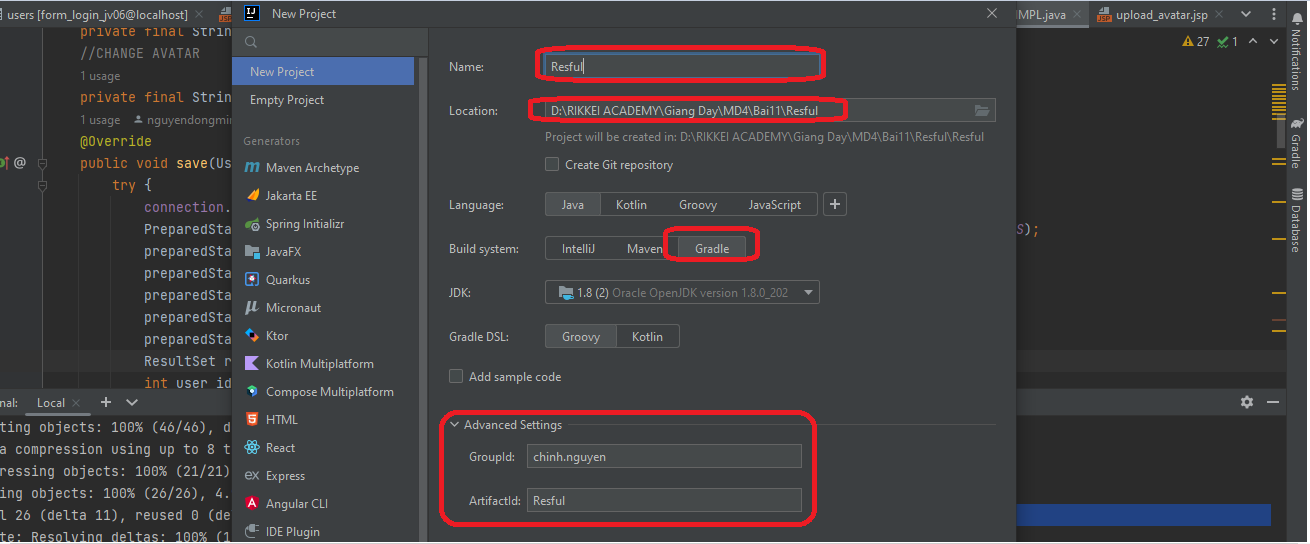
Sử dụng POSTMAN để kiểm tra hoạt động của các RESTful API.

Để hoàn thành bài thực hành, học viên cần:

* Đưa mã nguồn lên GitHub

Hướng dẫn

* Tạo dự án Spring MVC với gradle:



* Thêm thư viện vào: **build.gradle**

compileOnly('javax.servlet:javax.servlet-api:4.0.1')  
 implementation group: 'org.springframework', name: 'spring-core', version: '5.3.2'  
 implementation group: 'org.springframework', name: 'spring-context', version: '5.3.2'  
 implementation group: 'org.springframework', name: 'spring-beans', version: '5.3.2'  
 implementation group: 'org.springframework', name: 'spring-web', version: '5.3.2'  
 implementation group: 'org.springframework', name: 'spring-webmvc', version: '5.3.2'  
compileOnly('javax.servlet:javax.servlet-api:4.0.1')  
implementation group: 'org.springframework', name: 'spring-core', version: '5.3.2'  
implementation group: 'org.springframework', name: 'spring-context', version: '5.3.2'  
implementation group: 'org.springframework', name: 'spring-beans', version: '5.3.2'  
implementation group: 'org.springframework', name: 'spring-web', version: '5.3.2'  
implementation group: 'org.springframework', name: 'spring-webmvc', version: '5.3.2'  
implementation group: 'org.thymeleaf', name: 'thymeleaf-spring5', version: '3.0.11.RELEASE'  
implementation group: 'nz.net.ultraq.thymeleaf', name: 'thymeleaf-layout-dialect', version: '2.5.2'  
implementation group: 'org.hibernate', name: 'hibernate-core', version: '5.3.0.Final'  
implementation group: 'org.hibernate', name: 'hibernate-entitymanager', version: '5.3.0.Final'  
implementation group: 'org.springframework', name: 'spring-orm', version: '5.3.2'  
implementation group: 'mysql', name: 'mysql-connector-java', version: '8.0.22'  
implementation group: 'org.springframework.data', name: 'spring-data-jpa', version: '2.4.2'  
//Thư viện data json RESTFUL  
implementation group: 'com.fasterxml.jackson.core', name: 'jackson-databind', version: '2.12.1'

plugins **{** id 'java'  
 id 'war'  
**}**

* Tạo package: **chinh.nguyen.config =>** Tạo class: **AppConfig:**

package chinh.nguyen.config;  
  
import chinh.nguyen.service.CustomerServiceIMPL;  
import chinh.nguyen.service.ICustomerService;  
import org.springframework.beans.BeansException;  
import org.springframework.beans.factory.annotation.Qualifier;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.ApplicationContextAware;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.ComponentScan;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;  
import org.springframework.data.web.config.EnableSpringDataWebSupport;  
import org.springframework.jdbc.datasource.DriverManagerDataSource;  
import org.springframework.orm.jpa.JpaTransactionManager;  
import org.springframework.orm.jpa.JpaVendorAdapter;  
import org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean;  
import org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter;  
import org.springframework.transaction.PlatformTransactionManager;  
import org.springframework.transaction.annotation.EnableTransactionManagement;  
import org.springframework.web.servlet.config.annotation.EnableWebMvc;  
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;  
import org.thymeleaf.spring5.SpringTemplateEngine;  
import org.thymeleaf.spring5.templateresolver.SpringResourceTemplateResolver;  
import org.thymeleaf.spring5.view.ThymeleafViewResolver;  
import org.thymeleaf.templatemode.TemplateMode;  
  
import javax.persistence.EntityManager;  
import javax.persistence.EntityManagerFactory;  
import javax.sql.DataSource;  
import java.util.Properties;  
  
@Configuration  
@EnableWebMvc  
@EnableTransactionManagement  
@EnableSpringDataWebSupport  
@ComponentScan("chinh.nguyen.controller")  
@EnableJpaRepositories("chinh.nguyen.repository")  
public class AppConfig implements WebMvcConfigurer, ApplicationContextAware {  
 private ApplicationContext applicationContext;  
  
 @Override  
 public void setApplicationContext(ApplicationContext applicationContext) throws BeansException {  
 this.applicationContext = applicationContext;  
 }  
  
 //Cấu hình Thymleaf  
 @Bean  
 public SpringResourceTemplateResolver templateResolver() {  
 SpringResourceTemplateResolver templateResolver = new SpringResourceTemplateResolver();  
 templateResolver.setApplicationContext(applicationContext);  
 templateResolver.setPrefix("/WEB-INF/views");  
 templateResolver.setSuffix(".html");  
 templateResolver.setTemplateMode(TemplateMode.*HTML*);  
 templateResolver.setCharacterEncoding("UTF-8");  
 return templateResolver;  
 }  
  
 @Bean  
 public SpringTemplateEngine templateEngine() {  
 SpringTemplateEngine templateEngine = new SpringTemplateEngine();  
 templateEngine.setTemplateResolver(templateResolver());  
 return templateEngine;  
 }  
  
 @Bean  
 public ThymeleafViewResolver viewResolver() {  
 ThymeleafViewResolver viewResolver = new ThymeleafViewResolver();  
 viewResolver.setTemplateEngine(templateEngine());  
 viewResolver.setCharacterEncoding("UTF-8");  
 viewResolver.setContentType("UTF-8");  
 return viewResolver;  
 }  
  
 //Cấu hình JPA  
 @Bean  
 @Qualifier(value = "entityManager")  
 public EntityManager entityManager(EntityManagerFactory entityManagerFactory) {  
 return entityManagerFactory.createEntityManager();  
 }  
  
 @Bean  
 public LocalContainerEntityManagerFactoryBean entityManagerFactory() {  
 LocalContainerEntityManagerFactoryBean em = new LocalContainerEntityManagerFactoryBean();  
 em.setDataSource(dataSource());  
 em.setPackagesToScan("chinh.nguyen.model");  
  
 JpaVendorAdapter vendorAdapter = new HibernateJpaVendorAdapter();  
 em.setJpaVendorAdapter(vendorAdapter);  
 em.setJpaProperties(additionalProperties());  
 return em;  
 }  
  
 @Bean  
 public DataSource dataSource() {  
 DriverManagerDataSource dataSource = new DriverManagerDataSource();  
 dataSource.setDriverClassName("com.mysql.cj.jdbc.Driver");  
 dataSource.setUrl("jdbc:mysql://localhost:3306/bai11\_th\_restful");  
 dataSource.setUsername("root");  
 dataSource.setPassword("Minhtri29092014");  
 return dataSource;  
 }  
  
 @Bean  
 public PlatformTransactionManager transactionManager(EntityManagerFactory emf) {  
 JpaTransactionManager transactionManager = new JpaTransactionManager();  
 transactionManager.setEntityManagerFactory(emf);  
 return transactionManager;  
 }  
  
 public Properties additionalProperties() {  
 Properties properties = new Properties();  
 properties.setProperty("hibernate.hbm2ddl.auto", "update");  
 properties.setProperty("hibernate.dialect", "org.hibernate.dialect.MySQL5Dialect");  
 return properties;  
 }  
  
}

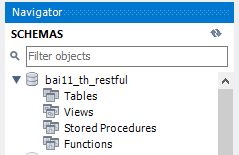
* Tạo class: **AppInit**

package chinh.nguyen.config;  
  
import org.springframework.web.filter.CharacterEncodingFilter;  
import org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitializer;  
  
import javax.servlet.Filter;  
  
public class AppInit extends AbstractAnnotationConfigDispatcherServletInitializer {  
 @Override  
 protected Class<?>[] getRootConfigClasses() {  
 return new Class[]{AppConfig.class};  
 }  
  
 @Override  
 protected Class<?>[] getServletConfigClasses() {  
 return new Class[0];  
 }  
  
 @Override  
 protected String[] getServletMappings() {  
 return new String[]{"/"};  
 }  
 @Override  
 protected Filter[] getServletFilters() {  
 CharacterEncodingFilter filter = new CharacterEncodingFilter();  
 filter.setForceEncoding(true);  
 filter.setEncoding("UTF-8");  
 return new Filter[]{filter};  
 }  
}

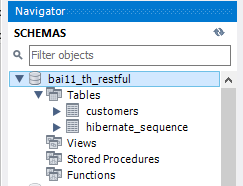
* Tạo package: **model** => Thêm vào class: **Customer**

package chinh.nguyen.model;  
  
import javax.persistence.\*;  
  
@Entity  
@Table(name = "customers")  
public class Customer {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*AUTO*)  
 private Long id;  
 private String firstName;  
 private String lastName;  
  
 public Customer(String firstName, String lastName) {  
 this.firstName = firstName;  
 this.lastName = lastName;  
 }  
  
 public Customer() {  
 }  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public void setFirstName(String firstName) {  
 this.firstName = firstName;  
 }  
  
 public String getLastName() {  
 return lastName;  
 }  
  
 public void setLastName(String lastName) {  
 this.lastName = lastName;  
 }  
  
}

* Tạo database: **bai11\_th\_restful** trong MySQL:



* Add Tomcat và chạy dự án quan sát trong database:



* Tạo package: **repository** => Thêm vào **interface**: **ICustomerRepository**:

package chinh.nguyen.repository;  
  
import chinh.nguyen.model.Customer;  
import org.springframework.data.repository.PagingAndSortingRepository;  
import org.springframework.stereotype.Repository;  
  
@Repository  
public interface ICustomerRepository extends PagingAndSortingRepository<Customer, Long> {  
}

* Tạo package: **service** => Thêm vào **interface**: **IGenericService:**

package chinh.nguyen.service;  
  
import java.util.Optional;  
  
public interface IGenericService <T>{  
 Iterable<T> findAll();  
  
 Optional<T> findById(Long id);  
  
 T save(T t);  
  
 void remove(Long id);  
  
}

* Tạo interface: **ICustomerService:**

package chinh.nguyen.service;  
  
import chinh.nguyen.model.Customer;  
  
public interface ICustomerService extends IGenericService<Customer> {  
}

* Tạo class: **CustomerServiceIMPL**:

package chinh.nguyen.service;  
  
import chinh.nguyen.model.Customer;  
import chinh.nguyen.repository.ICustomerRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.Optional;  
@Service  
public class CustomerServiceIMPL implements ICustomerService{  
 @Autowired  
 private ICustomerRepository customerRepository;  
  
 @Override  
 public Iterable<Customer> findAll() {  
 return customerRepository.findAll();  
 }  
  
 @Override  
 public Optional<Customer> findById(Long id) {  
 return customerRepository.findById(id);  
 }  
  
 @Override  
 public Customer save(Customer customer) {  
 return customerRepository.save(customer);  
 }  
  
 @Override  
 public void remove(Long id) {  
 customerRepository.deleteById(id);  
 }  
  
}

* Tạo package: controller **CustomerController**

Sau đây là controller dựa trên REST, thi hành REST API.

* GET request "/api/customers/" trả về danh sách các khách hàng
* GET request "/api/customers/1" trả về khách hàng với ID 1
* POST request "/api/customers/" với một JSON object tạo một khách hàng mới
* PUT request "/api/customers/3" với một JSON object cập nhật khách hàng có ID 3
* DELETE request "/api/customers/3" xóa khách hàng có ID 3

package chinh.nguyen.controller;  
  
import chinh.nguyen.model.Customer;  
import chinh.nguyen.service.ICustomerService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.stereotype.Controller;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
import java.util.Optional;  
  
@RestController  
@RequestMapping("/api/customers")  
public class CustomerController {  
  
 @Autowired  
 private ICustomerService customerService;  
  
 @GetMapping  
 public ResponseEntity<Iterable<Customer>> findAllCustomer() {  
 List<Customer> customers = (List<Customer>) customerService.findAll();  
 if (customers.isEmpty()) {  
 return new ResponseEntity<>(HttpStatus.*NO\_CONTENT*);  
 }  
 return new ResponseEntity<>(customers, HttpStatus.*OK*);  
 }  
  
 @GetMapping("/{id}")  
 public ResponseEntity<Customer> findCustomerById(@PathVariable Long id) {  
 Optional<Customer> customerOptional = customerService.findById(id);  
 if (!customerOptional.isPresent()) {  
 return new ResponseEntity<>(HttpStatus.*NOT\_FOUND*);  
 }  
 return new ResponseEntity<>(customerOptional.get(), HttpStatus.*OK*);  
 }  
  
 @PostMapping  
 public ResponseEntity<Customer> saveCustomer(@RequestBody Customer customer) {  
 return new ResponseEntity<>(customerService.save(customer), HttpStatus.*CREATED*);  
 }  
  
 @PutMapping("/{id}")  
 public ResponseEntity<Customer> updateCustomer(@PathVariable Long id, @RequestBody Customer customer) {  
 Optional<Customer> customerOptional = customerService.findById(id);  
 if (!customerOptional.isPresent()) {  
 return new ResponseEntity<>(HttpStatus.*NOT\_FOUND*);  
 }  
 customer.setId(customerOptional.get().getId());  
 return new ResponseEntity<>(customerService.save(customer), HttpStatus.*OK*);  
 }  
  
 @DeleteMapping("/{id}")  
 public ResponseEntity<Customer> deleteCustomer(@PathVariable Long id) {  
 Optional<Customer> customerOptional = customerService.findById(id);  
 if (!customerOptional.isPresent()) {  
 return new ResponseEntity<>(HttpStatus.*NOT\_FOUND*);  
 }  
 customerService.remove(id);  
 return new ResponseEntity<>(customerOptional.get(), HttpStatus.*NO\_CONTENT*);  
 }  
  
}

Trong đó:

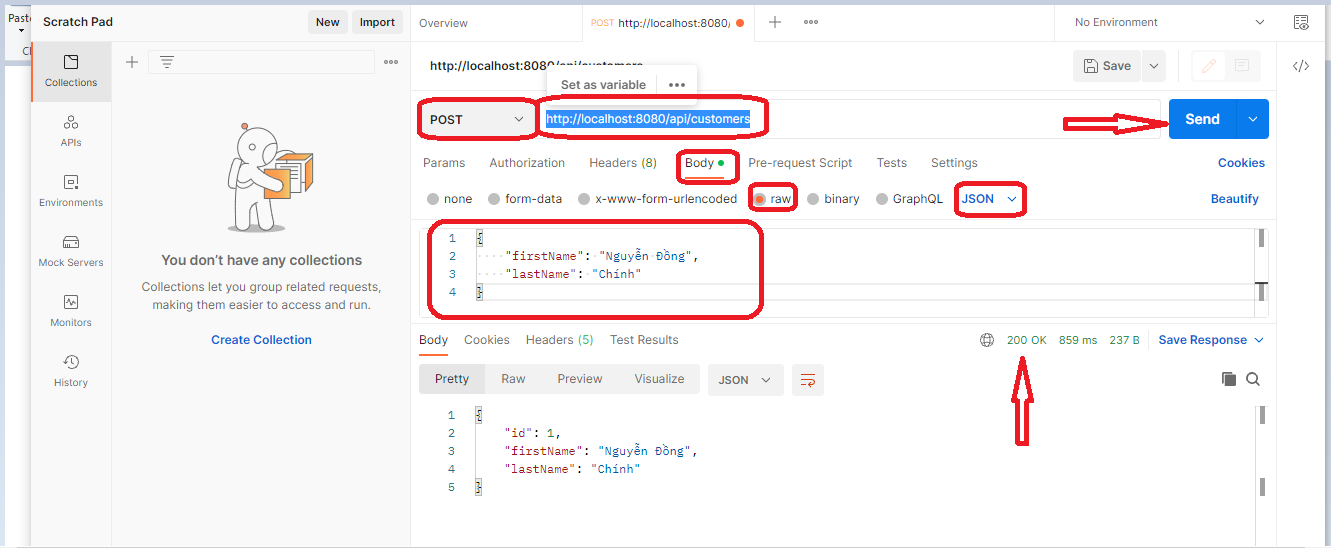
* @RestController là kết hợp của @Controller và @ResponseBody
* @RequestBody: Nếu tham số phương thức được chú thích bằng @RequestBody, Spring sẽ liên kết phần thân yêu cầu HTTP đến với tham số đó.
* ResponseEntity đại diện cho toàn bộ phản hồi HTTP
* @PathVariable chỉ ra rằng tham số phương thức sẽ được liên kết với URI (id)
* Thêm @Bean lớp service vào **AppConfig:**

@Bean  
public ICustomerService customerService(){  
 return new CustomerServiceIMPL();  
}

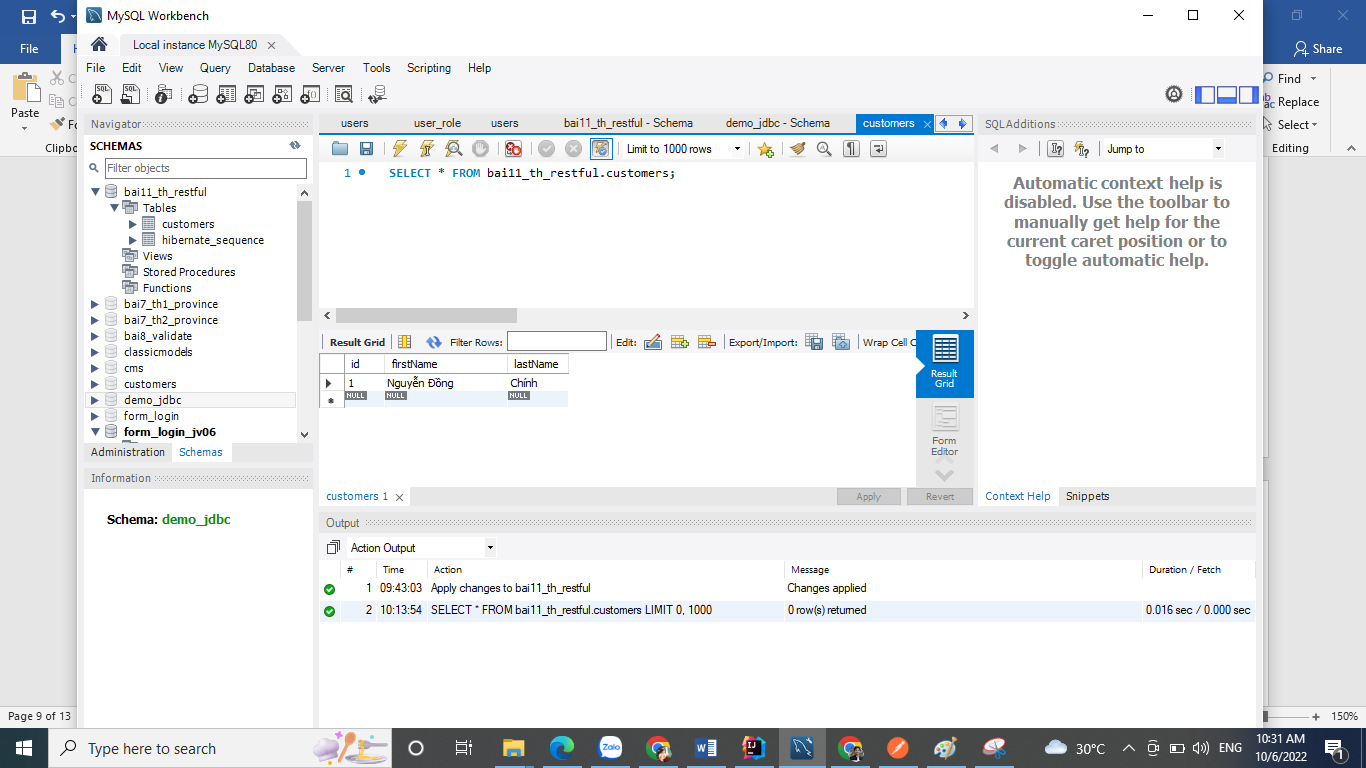
* **Triển khai và test API**

Để test API, chúng ta sử dụng POSTMAN

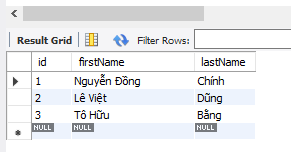
1. Test chức năng Create với API: <http://localhost:8080/api/customers>



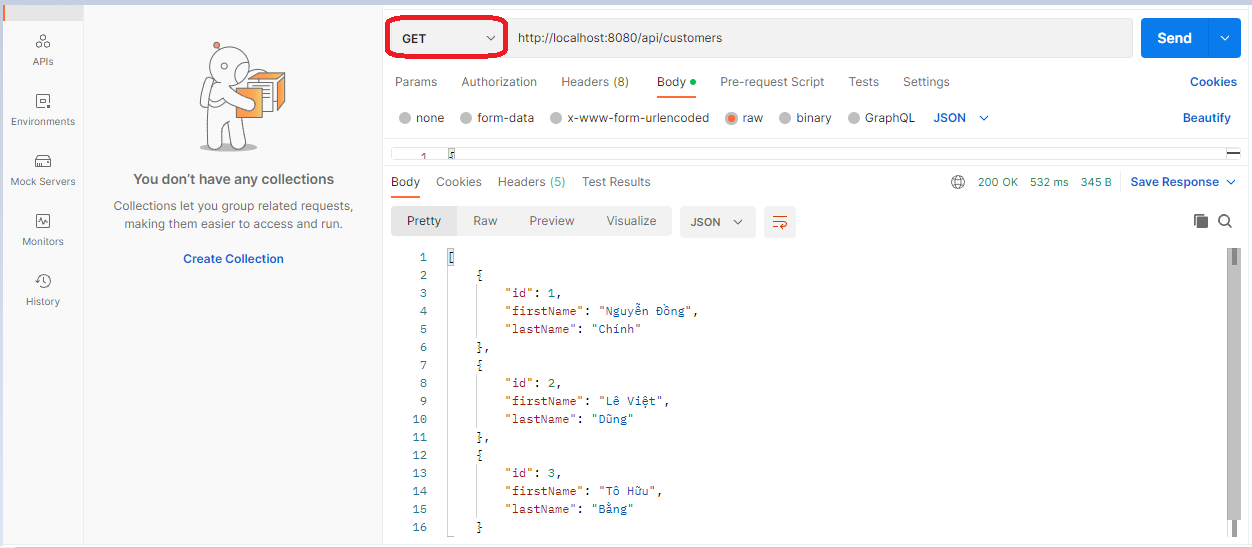
* Kiểm tra database:



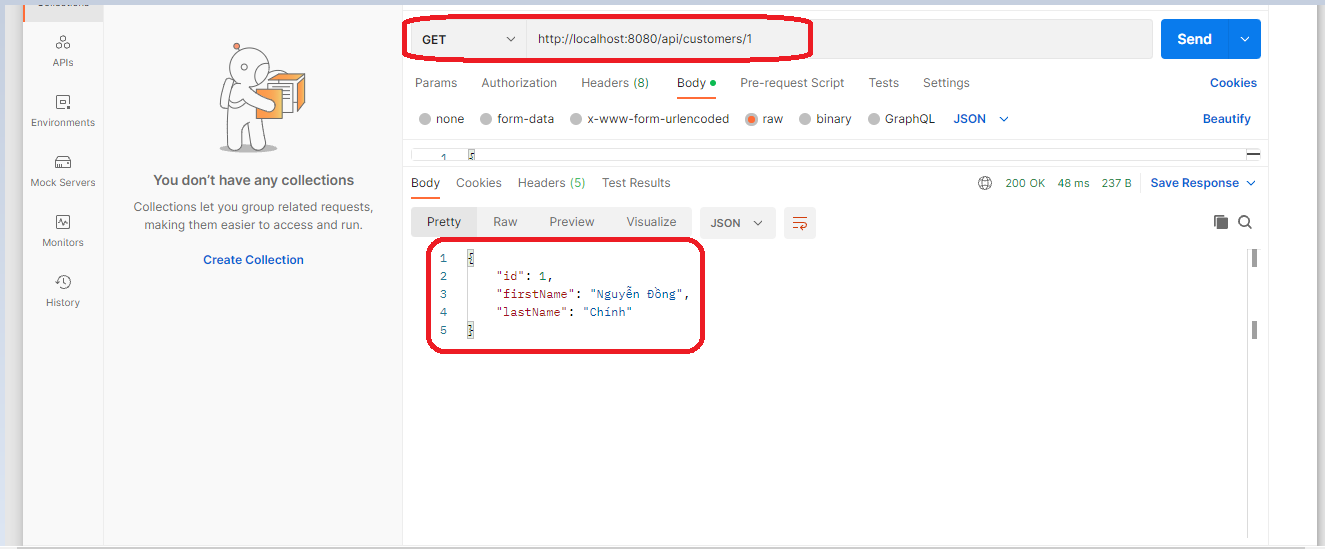
* Thử thêm một vài đối tượng **customer** khác bằng POSTMAN



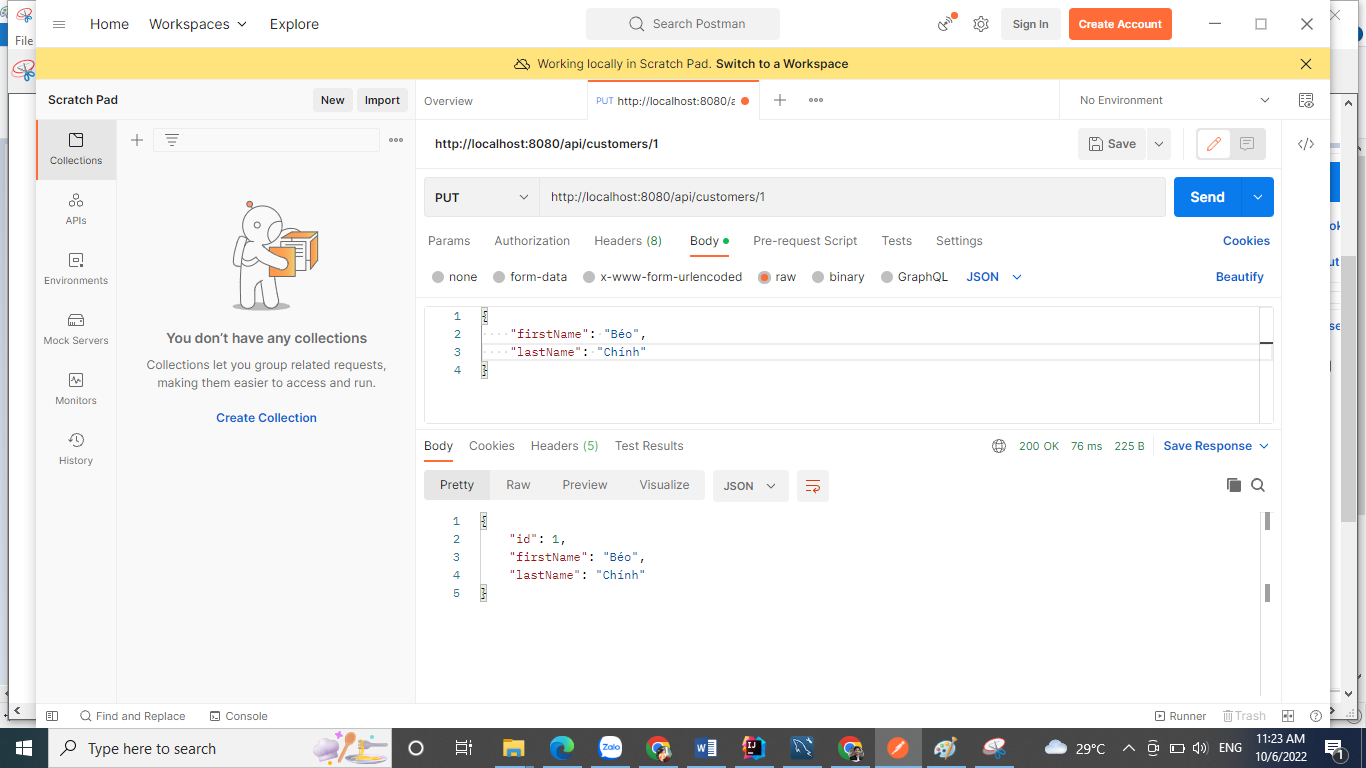
1. Test chức năng show list bằng POSTMAN vẫn API: <http://localhost:8080/api/customers> nhưng đổi sang method GET



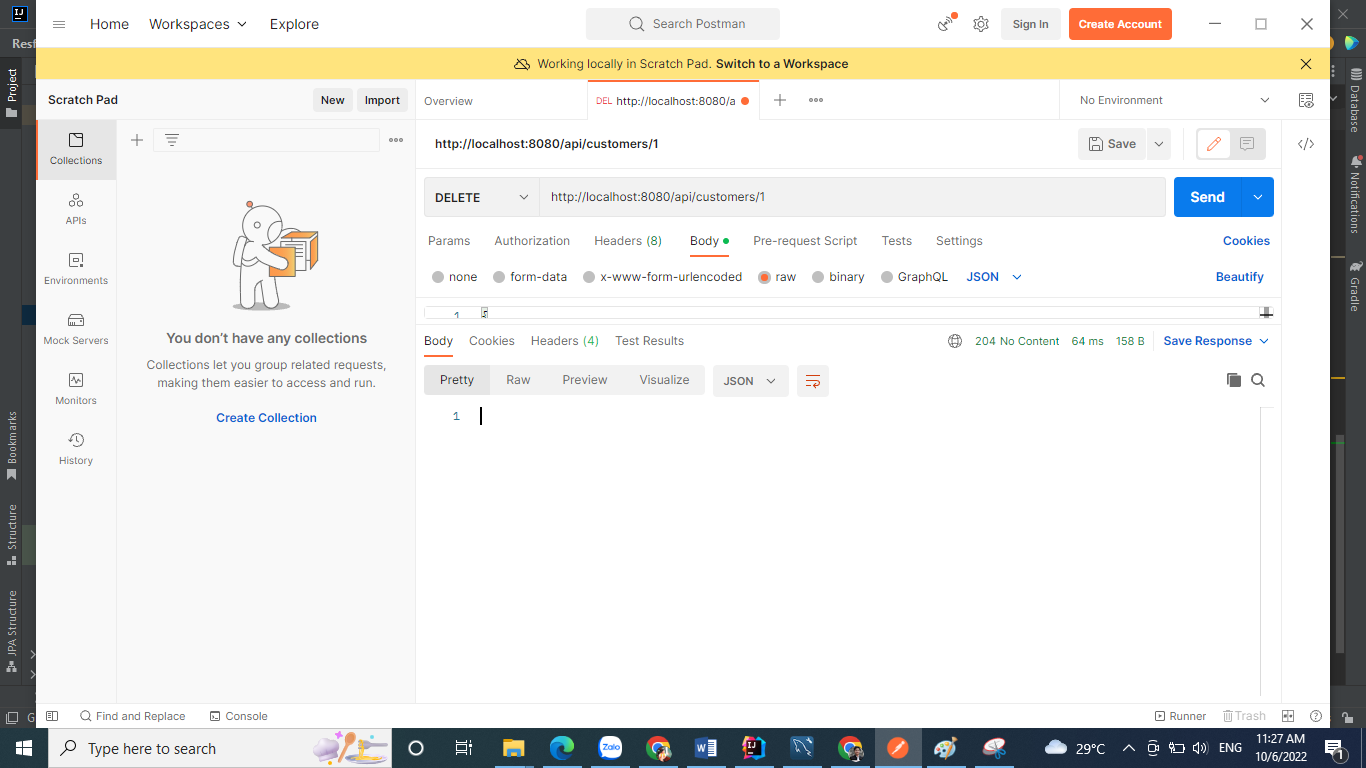
**3.** Hiển thị khách hàng theo ID: API lấy ra id/1 : http://localhost:8080/api/customers/1



* **4.**Sửa thông tin khách hàng: API sẳ id/1: (Method PUT): http://localhost:8080/api/customers/1



**5.** Xóa khách hàng: API xóa id/1: (Method: DELETE) http://localhost:8080/api/customers/1



### <https://github.com/nguyendongminhtri/MD4-Bai11-TH-RestFul-SpringMVC.git>

### Hướng dẫn nộp bài:

Up code lên github.

Paste link github vào phần nộp bài.