[Thực hành] Ứng dụng quản lý danh sách khách hàng

Mục tiêu

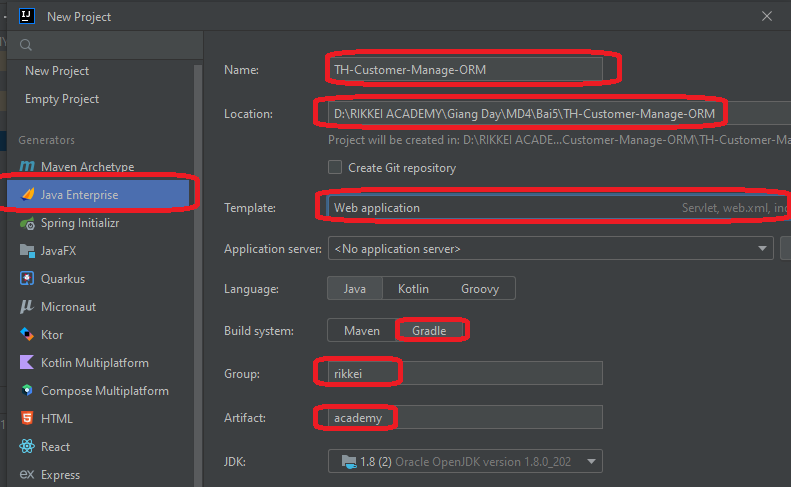
* ***Luyện tập tạo config dự án bằng JAVA – view Thyme-Leaf – Kết hợp ánh frameword ORM phổ biến là hibernate để ánh xạ bộ nhớ của chương trình với database***

Mô tả

Bổ sung khả năng ngoại hóa cơ sở dữ liệu cho chương trình quản lý danh sách khách hàng.

Hướng dẫn

* Tạo dự án mới như sau:



* Thêm thư viện vào file: **build.gradle** như sau:

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'

* **mô tả entity**

Mô tả dữ liệu cần ngoại hóa như là các entity, mỗi entity như một dòng dữ liệu trong cơ sở dữ liệu quan hệ. Mỗi trường của entity như giá trị tại một cột tương ứng của dòng dữ liệu đó.

Với Hibernate, việc mô tả một entity được thực hiện trên class, việc mô tả một class tương ứng với một entity được thực hiện bởi hành động đánh annotation cho class và cho các trường của class đó.

Trước tiên, bổ sung thư viện Hibernate => build.gradle

implementation group: 'org.hibernate', name: 'hibernate-core', version: '5.4.3.Final'

Đánh dấu rằng class **Customer** => trong package model: là tương ứng với một entity:

import javax.persistence.\*;

@Entity  
@Table(name = "customers")  
public class Customer {  
 // ...  
}

Hibernate cần biết cách gắn một đối tượng của class customer với một dòng dữ liệu cụ thể tại database, do đó nó cần biết trường nào trong class Customer là tương ứng với khóa chính:

@Id  
@GeneratedValue(strategy = GenerationType.*IDENTITY*)  
private Long id;

* **Toàn bộ: class: Customer lúc này sẽ như sau:**

package rikkei.academy.model;  
  
import javax.persistence.\*;  
  
@Entity  
@Table(name = "customers")  
public class Customer {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
 private String name;  
 private String email;  
 private String address;  
  
 public Customer() {  
 }  
  
 public Customer(String name, String email, String address) {  
 this.name = name;  
 this.email = email;  
 this.address = address;  
 }  
  
 public Customer(Long id, String name, String email, String address) {  
 this.id = id;  
 this.name = name;  
 this.email = email;  
 this.address = address;  
 }  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
  
 public String getAddress() {  
 return address;  
 }  
  
 public void setAddress(String address) {  
 this.address = address;  
 }  
  
 @Override  
 public Customer clone() {  
 Customer customer = new Customer();  
 customer.setId(id);  
 customer.setName(name);  
 customer.setEmail(email);  
 customer.setAddress(address);  
 return customer;  
 }  
  
 @Override  
 public String toString() {  
 return "Customer{" +  
 "id=" + id +  
 ", name='" + name + '\'' +  
 ", email='" + email + '\'' +  
 ", address='" + address + '\'' +  
 '}';  
 }  
}

* **Tạo package: service => Tạo interface: ICustomerService => Thêm code vào như sau:**

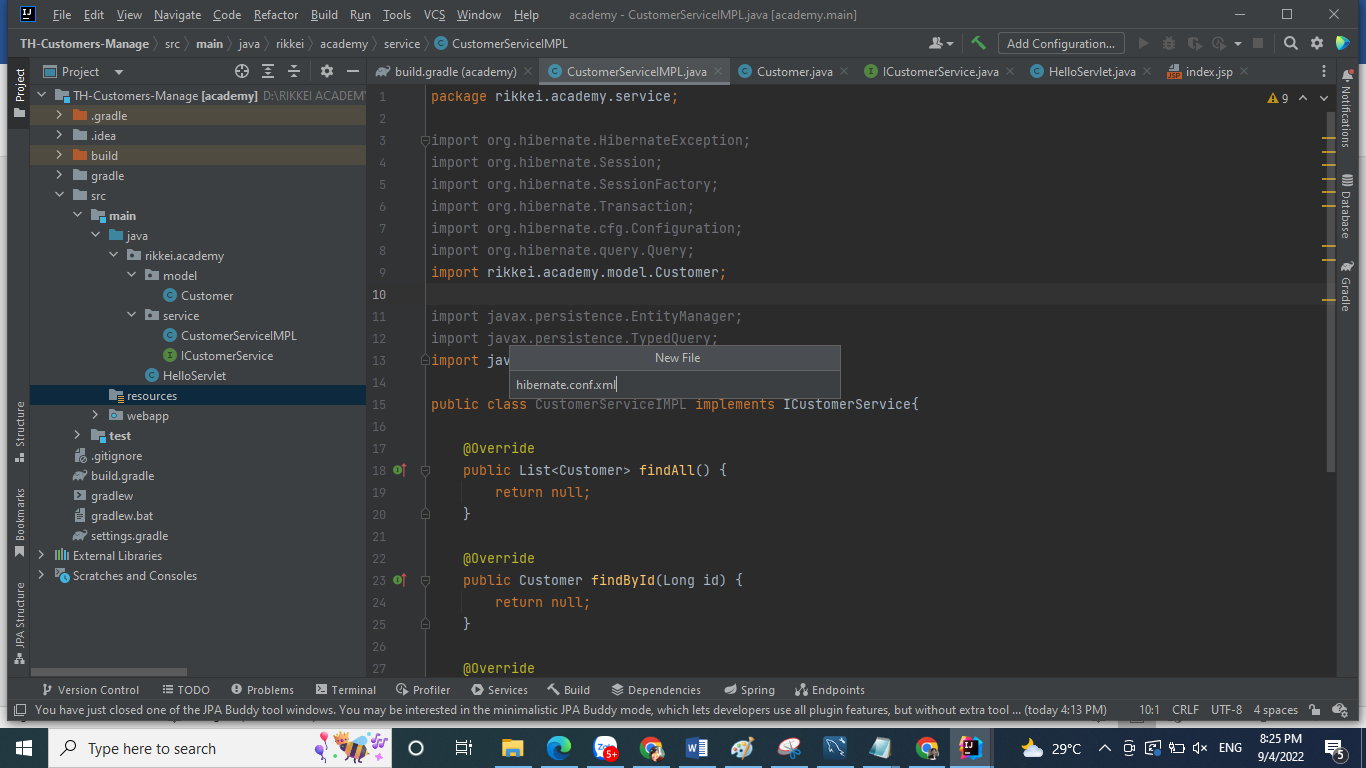
package rikkei.academy.service;  
  
import rikkei.academy.model.Customer;  
  
import java.util.List;  
  
public interface ICustomerService {  
 List<Customer> findAll();  
  
 Customer findById(Long id);  
  
 void deleteById(Long id);  
  
 void save(Customer customer);  
}

* **Thêm thư viện vào: build.gradle**

implementation group: 'mysql', name: 'mysql-connector-java', version: '8.0.11'

**Thư viện này cài đặt Driver cho MySQL**

* **Tạo file: hibernate.conf.xml trong thư mục: resources**



* **Thêm code config vào như sau:**

<?xml version="1.0" encoding="utf-8"?>  
<!DOCTYPE hibernate-configuration PUBLIC  
 "-//Hibernate/Hibernate Configuration DTD 3.0//EN"  
 "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">  
<hibernate-configuration>  
 <session-factory>  
 <property name="hibernate.connection.driver\_class">com.mysql.jdbc.Driver</property>  
 <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/cms?useSSL=false</property>  
 <property name="hibernate.connection.username">root</property>  
 <property name="hibernate.connection.password">Minhtri29092014</property>  
 <property name="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect</property>  
 <property name="show\_sql">true</property>  
 <property name="hbm2ddl.auto">update</property>  
 <mapping class="rikkei.academy.model.Customer"/>  
 </session-factory>  
</hibernate-configuration>

* "show\_sql" gợi ý rằng các câu SQL mà Hibernate sử dụng sẽ được log ra logstream của hệ thống, "hbm2ddl.auto" gợi ý rằng khi database không có bảng, hay các bảng trong database cũ hơn so với mô tả tại entity class, hibernate sẽ tiến hành cập nhật cấu trúc các bảng.  
  Trong jdbc URL, cms chính là tên của database
* Tạo database có tên : **cms** trong phần mềm MySQL Workbench => điền mật khẩu MySQL tương ứng của các bạn ở dòng password => Sửa phần mapping xuống package model chứa Customer:

<mapping class="rikkei.academy.model.Customer"/>

* Tạo class: **CustomerServiceIMPL** triển khai **IcustomerService:** Thêm vào đoạn code sau ngoài các phương thức triển khai

private static SessionFactory *sessionFactory*;  
private static EntityManager *entityManager*;  
static {  
 try {  
 *sessionFactory* = new Configuration()  
 .configure("hibernate.conf.xml")  
 .buildSessionFactory();  
 *entityManager* = *sessionFactory*.createEntityManager();  
 } catch (HibernateException e) {  
 e.printStackTrace();  
 }  
}

* Class: **CustomerServiceIMPL** lúc này như sau:

package rikkei.academy.service;  
import org.hibernate.HibernateException;  
import org.hibernate.SessionFactory;  
import org.hibernate.cfg.Configuration;  
import rikkei.academy.model.Customer;  
  
import javax.persistence.EntityManager;  
import java.util.List;  
public class CustomerServiceIMPL implements ICustomerService{  
 private static SessionFactory *sessionFactory*;  
 private static EntityManager *entityManager*;  
 static {  
 try {  
 *sessionFactory* = new Configuration()  
 .configure("hibernate.conf.xml")  
 .buildSessionFactory();  
 *entityManager* = *sessionFactory*.createEntityManager();  
 } catch (HibernateException e) {  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 public List<Customer> findAll() {  
 return null;  
 }  
  
 @Override  
 public Customer findById(Long id) {  
 return null;  
 }  
  
 @Override  
 public void deleteById(Long id) {  
  
 }  
  
 @Override  
 public void save(Customer customer) {  
  
 }  
}

* **Các bạn nhớ lưu ý import các thư viện cho đúng** SessionFactory vs EntityManager
* Tạo package: config => Thêm các class: **AppInit** vs **AppConfig** như sau:
* **AppInit:**

package rikkei.academy.config;  
  
import org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitializer;  
  
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[]{"/"};  
 }  
}

* **AppConfig:**

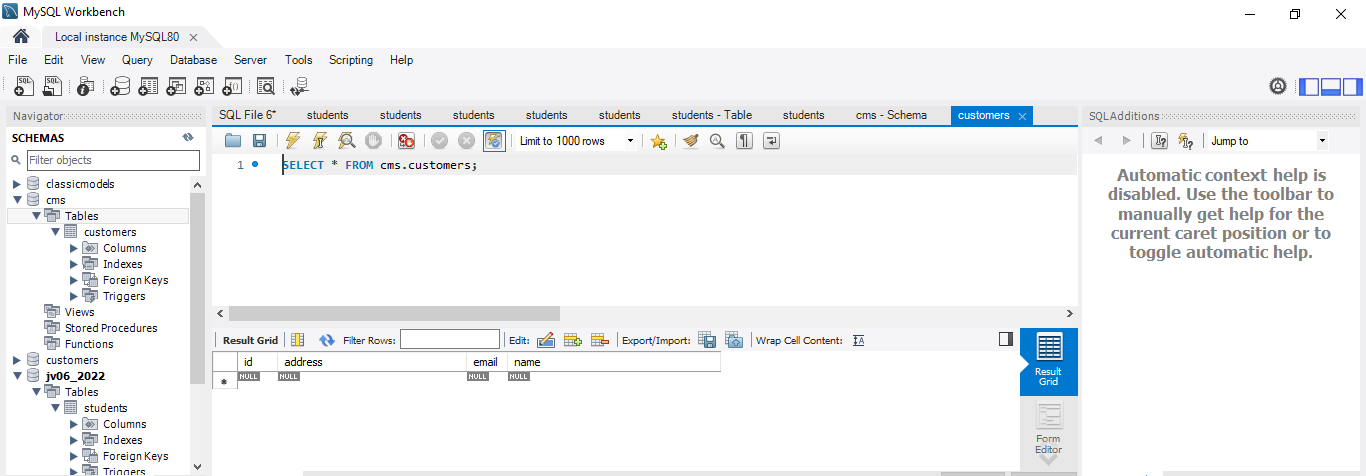
package rikkei.academy.config;  
  
import org.springframework.beans.BeansException;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.ApplicationContextAware;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.ComponentScan;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.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 rikkei.academy.service.CustomerServiceIMPL;  
import rikkei.academy.service.ICustomerService;  
  
@Configuration  
@EnableWebMvc  
@ComponentScan("rikkei.academy.controller")  
public class AppConfig implements WebMvcConfigurer, ApplicationContextAware{  
 private ApplicationContext appContext;  
  
 @Override  
 public void setApplicationContext(ApplicationContext applicationContext) throws BeansException {  
 appContext = applicationContext;  
 }  
  
 @Bean  
 public ThymeleafViewResolver viewResolver() {  
 ThymeleafViewResolver viewResolver = new ThymeleafViewResolver();  
 viewResolver.setTemplateEngine(templateEngine());  
 return viewResolver;  
 }  
  
 @Bean  
 public SpringTemplateEngine templateEngine() {  
 SpringTemplateEngine templateEngine = new SpringTemplateEngine();  
 templateEngine.setTemplateResolver(templateResolver());  
 return templateEngine;  
 }  
  
 @Bean  
 public SpringResourceTemplateResolver templateResolver() {  
 SpringResourceTemplateResolver templateResolver = new SpringResourceTemplateResolver();  
 templateResolver.setApplicationContext(appContext);  
 templateResolver.setPrefix("/WEB-INF/views/");  
 templateResolver.setSuffix(".html");  
 templateResolver.setTemplateMode(TemplateMode.*HTML*);  
 return templateResolver;  
 }  
 @Bean  
 public ICustomerService customerService(){  
 return new CustomerServiceIMPL();  
 }  
}

* **Lưu ý:**

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

Đoạn code này khởi tạo tiêm sự phụ thuộc khởi chạy lớp CustomerServiceIMPL

* **Tạo thêm package: controller =>** @ComponentScan("rikkei.academy.controller")**)**
* Tạo **cms** trong MySQL workbench
* Add Tomcat xóa các file cũ của Java Servlet lúc khởi tạo dự án:class: **HelloServlet, index.jsp** và chạy chương trình quan sát dưới cơ sở dữ liệu: MySQL



* **Ta thấy việc kết nối giữa Model: Customer thông qua hibernate đã tạo được bảng Database tương ứng dưới CSDL.**
* **Xây dựng việc thêm dữ liệu và show ra list : Customer => Sửa lại code CustomerService như sau:**
* **=>** Phương thức findAll():

@Override  
public List<Customer> findAll() {  
 String queryStr = "SELECT c FROM Customer AS c";  
 TypedQuery<Customer> query = *entityManager*.createQuery(queryStr, Customer.class);  
 return query.getResultList();  
}

* **=>** Phương thức: **save():**
* @Override  
  public void save(Customer customer) {  
   Session session = null;  
   Transaction transaction = null;  
   try {  
   session = *sessionFactory*.openSession();  
   transaction = session.beginTransaction();  
   session.saveOrUpdate(customer);  
   transaction.commit();  
   } catch (Exception e) {  
   e.printStackTrace();  
   if (transaction != null) {  
   transaction.isActive();  
   }  
   } finally {  
   if (session != null) {  
   session.close();  
   }  
   }  
  }
* Tạo class: **CustomerController** trong package: **controller =>** Thêm code vào như sau:

@Autowired  
private ICustomerService customerService;  
@GetMapping("/")  
public ModelAndView listCustomers(){  
 ModelAndView modelAndView = new ModelAndView("/customer/list");  
 modelAndView.addObject("customers", customerService.findAll());  
 return modelAndView;  
}  
@GetMapping("/create")  
public String showFormCreate(Model model) {  
 Customer customer = new Customer();  
 model.addAttribute("customerForm",customer);  
 System.*out*.println("name"+customer.getName());  
 return "/customer/add";  
}  
  
@PostMapping("/create/customer")  
public String createCustomer(@ModelAttribute("customerForm") Customer customer) {  
 customerService.save(customer);  
 return "redirect:/";  
}

* Class: **CustomerController** lúc này như sau:

package rikkei.academy.controller;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.ModelAttribute;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.servlet.ModelAndView;  
import rikkei.academy.model.Customer;  
import rikkei.academy.service.ICustomerService;  
  
@Controller  
public class CustomerController {  
 @Autowired  
 private ICustomerService customerService;  
 @GetMapping("/")  
 public ModelAndView listCustomers(){  
 ModelAndView modelAndView = new ModelAndView("/customer/list");  
 modelAndView.addObject("customers", customerService.findAll());  
 return modelAndView;  
 }  
 @GetMapping("/create")  
 public String showFormCreate(Model model) {  
 Customer customer = new Customer();  
 model.addAttribute("customerForm",customer);  
 System.*out*.println("name"+customer.getName());  
 return "/customer/add";  
 }  
  
 @PostMapping("/create/customer")  
 public String createCustomer(@ModelAttribute("customerForm") Customer customer) {  
 customerService.save(customer);  
 return "redirect:/";  
 }  
  
}

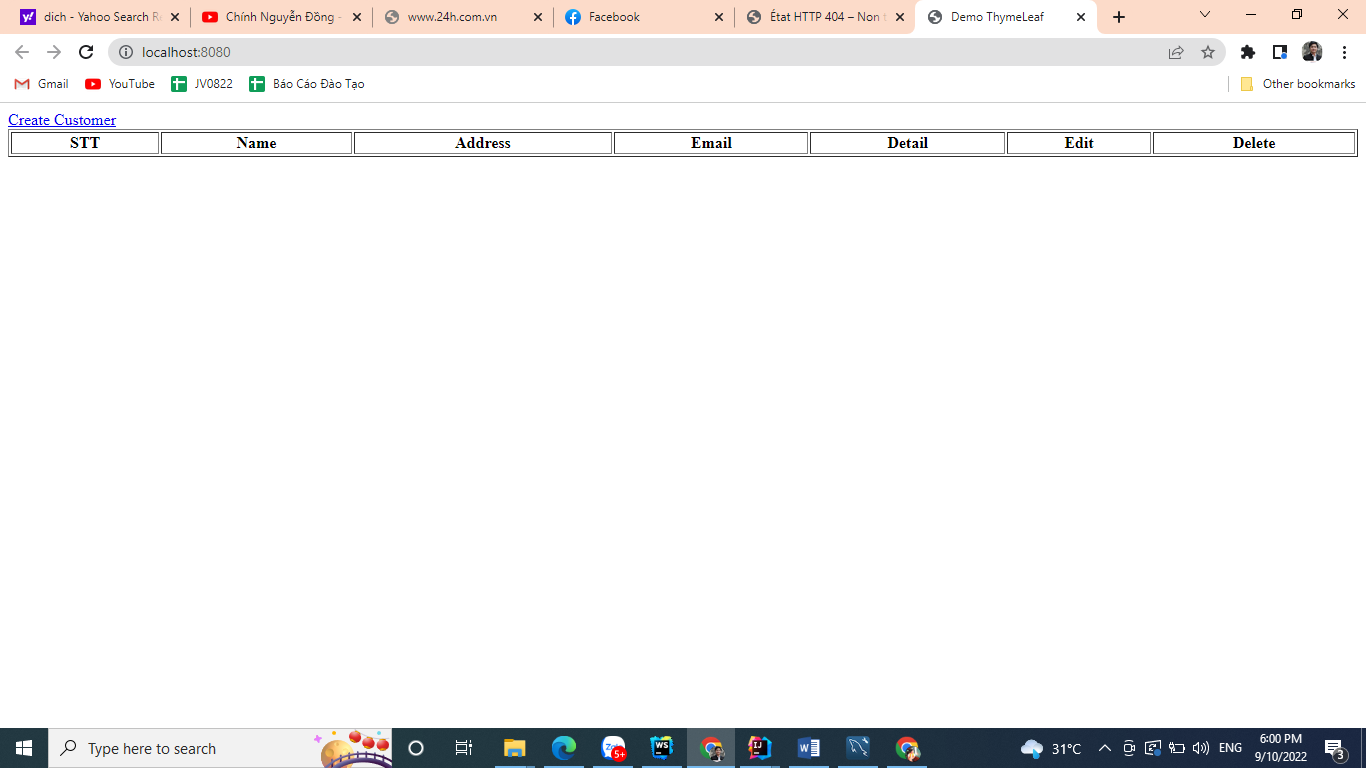
* Tạo tầng:  **views** (WEB-INF/views/customer) => Tạo file: **list.html** và thêm code vào như sau:

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Demo ThymeLeaf</title>  
</head>  
<style>  
 table {  
 width: 100%;  
 }  
</style>  
<body>  
  
<!--<h1>Hello C1220G1! We will be the developers of the future! </h1>-->  
<!--<h1 th:utext="${name}"></h1>-->  
<a href="/create">Create Customer</a>  
<table border="1">  
 <tr>  
 <th>STT</th>  
 <th>Name</th>  
 <th>Address</th>  
 <th>Email</th>  
 <th>Detail</th>  
 <th>Edit</th>  
 <th>Delete</th>  
 </tr>  
 <tr th:each="customer:${customers}">  
 <td th:utext="${customer.getId()}"></td>  
 <td th:utext="${customer.getName()}"></td>  
 <td th:utext="${customer.getAddress()}"></td>  
 <td th:utext="${customer.getEmail()}"></td>  
 <td><a th:href="@{/showForm/{id}(id=${customer.getId()})}">Detail</a> </td>  
 <td><a th:href="@{/edit/{id}(id=${customer.getId()})}">Edit</a></td>  
 <td><a th:href="@{/delete/{id}(id=${customer.getId()})}">Delete</a> </td>  
 </tr>  
  
</table>  
</body>  
</html>

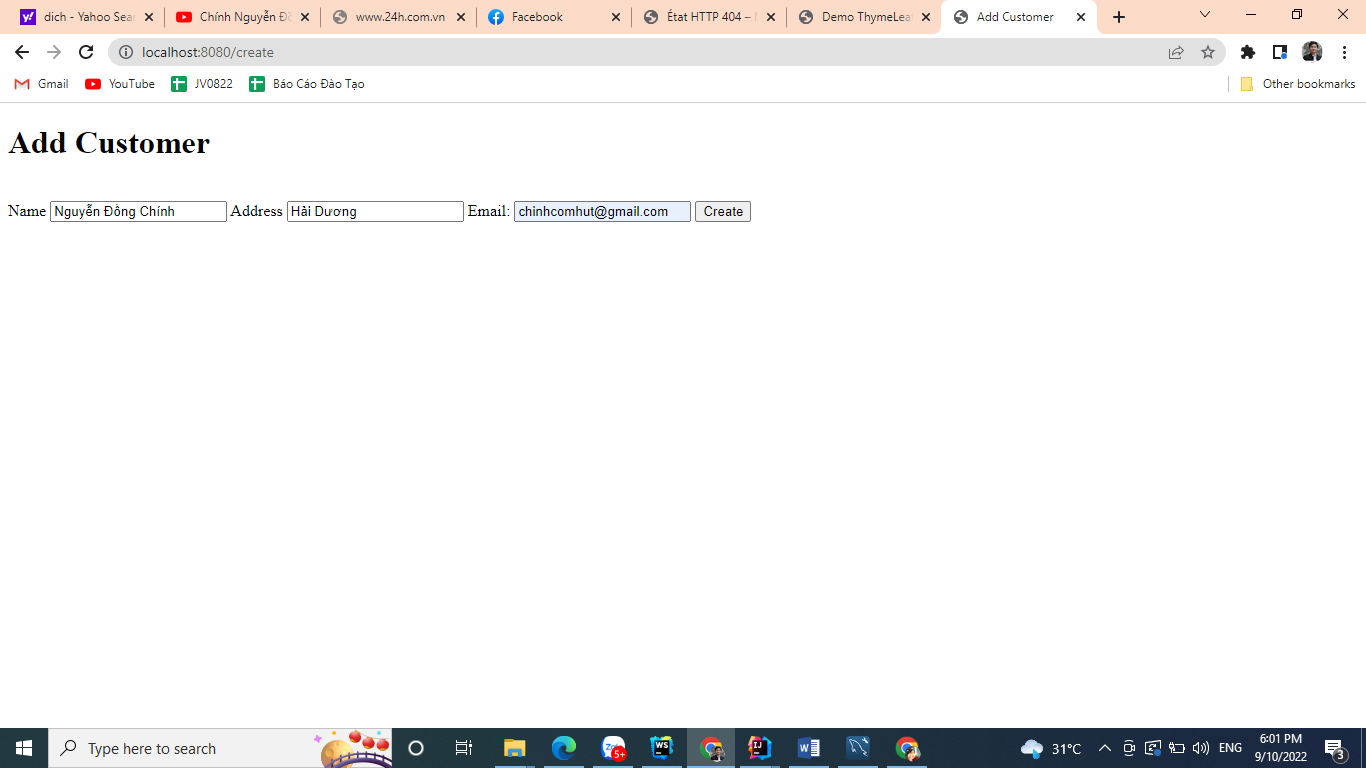
* Tạo file: **add.html** và thêm code vào như sau:

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Add Customer</title>  
</head>  
<body>  
<h1>Add Customer</h1><br>  
<form action="/create/customer" method="post" th:object="${customerForm}">  
 <label>Name</label>  
 <input type="text" name="name" th:field="\*{name}">  
 <label>Address</label>  
 <input type="text" th:field="\*{address}">  
 <lable>Email: </lable>  
 <input type="text" th:field="\*{email}">  
 <button type="submit">Create</button>  
</form>  
</body>  
</html>

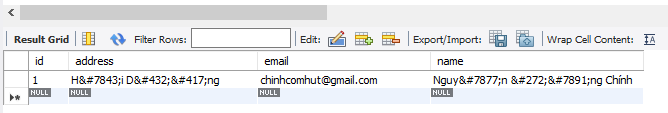
* Chạy lại trương trình và quan sát kết quả:



* Thêm vào dữ liệu và check dưới database



* Kết quả:



* Xây dựng chứng năng : **detail**
* Thêm code vào **CustomerController**:

@GetMapping("/detail/{id}")  
public String showFormForAdd(@PathVariable("id") Long id, Model theModel) {  
 System.*out*.println("id ==== "+id);  
 Customer theCustomer = customerService.findById(id);  
 theModel.addAttribute("customer", theCustomer);  
 return "customer/detail";  
}

* Sửa lại phương thức: **findById()** trong **CustomerServiceIMPL**

@Override  
public Customer findById(Long id) {  
 String queryStr = "SELECT c FROM Customer AS c WHERE c.id = :id";  
 TypedQuery<Customer> query = *entityManager*.createQuery(queryStr, Customer.class);  
 query.setParameter("id", id);  
 return query.getSingleResult();  
}

* Tạo file: **detail.html** trong thư mục …views/customer và thêm code vào như sau:

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
</head>  
<body>  
<table border="1">  
 <tr>  
 <th>STT</th>  
 <th>Name</th>  
 <th>Address</th>  
 <th>Email</th>  
 </tr>  
 <tr th:object="${customer}">  
 <td th:utext="${customer.getId()}"></td>  
 <td th:utext="${customer.getName()}"></td>  
 <td th:utext="${customer.getAddress()}"></td>  
 <td th:utext="${customer.getEmail()}"></td>  
 </tr>  
</table>  
</body>  
</html>

* Chạy lại phần mềm nhấn vào mục detail => quan sát kết quả
* **Xây dựng chức năng delete Customer:**
* Thêm code vào **CustomerController** như sau:

@GetMapping("/delete/{id}")  
public ModelAndView showFormDelete(@PathVariable Long id){  
 Customer customer = customerService.findById(id);  
 ModelAndView modelAndView = new ModelAndView("customer/delete");  
 modelAndView.addObject("deleteForm",customer);  
 return modelAndView;  
}  
@PostMapping("/delete")  
public String deleteById(@ModelAttribute("deleteForm") Customer customer) {  
 customerService.deleteById(customer.getId());  
 return "redirect:/";  
}

* Sửa lại phương thức: **deleteById()** trong **CustomerServiceIMPL :**

@Override  
public void deleteById(Long id) {  
 Session session = null;  
 Transaction transaction = null;  
 try {  
 session = *sessionFactory*.openSession();  
 transaction = session.beginTransaction();  
 session.delete(findById(id));  
 transaction.commit();  
 } catch (Exception e) {  
 e.printStackTrace();  
 if (transaction != null) {  
 transaction.rollback();  
 }  
 } finally {  
 if (session != null) {  
 session.close();  
 }  
 }  
}

* Tạo thư mục: **delete.html** trong **… views/customer:** Thêm code vào như sau:

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Delete</title>  
</head>  
<body>  
<h1>Are you sure delete this customer?</h1>  
<form action="/delete/" method="post" th:object="${deleteForm}">  
 <input type="text" name="id" th:value="\*{id}" hidden>  
 <h1 th:utext="${deleteForm.name}"></h1>  
 <button type="submit">Delete</button>  
</form>  
</body>  
</html>

* Chạy lại sever và test chức năng xóa check dưới database.
* **Xây dựng chức năng update**
* Thêm code vào **CustomerController:**

@GetMapping("/edit/{id}")  
public ModelAndView showFormEdit(@PathVariable Long id){  
 Customer customer = customerService.findById(id);  
 ModelAndView modelAndView = new ModelAndView("customer/edit");  
 modelAndView.addObject("editForm",customer);  
 return modelAndView;  
}  
@PostMapping("/edit/customer")  
public String editCustomer(@ModelAttribute("editForm") Customer customer){  
 customerService.save(customer);  
 return "redirect:/";  
}

* Sửa code phương thức **save()** trong **CustomerServiceIMPL**

@Override  
public void save(Customer customer) {  
 Session session = null;  
 Transaction transaction = null;  
 try {  
 session = *sessionFactory*.openSession();  
 transaction = session.beginTransaction();  
 if (customer.getId() != null) {  
 Customer customer1 = findById(customer.getId());  
 customer1.setName(customer.getName());  
 customer1.setEmail(customer.getEmail());  
 customer1.setAddress(customer.getAddress());  
 }  
 session.saveOrUpdate(customer);  
 transaction.commit();  
  
 } catch (Exception e) {  
 e.printStackTrace();  
 if (transaction != null) {  
 transaction.rollback();  
 }  
 } finally {  
 if (session != null) {  
 session.close();  
 }  
 }  
}

* Tạo file**: edit.html** trong **…views/customer:**

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Edit Customer</title>  
</head>  
<body>  
<form action="/edit/customer" method="post" th:object="${editForm}">  
 <input type="hidden" name="id" th:field="\*{id}">  
 <label>Name</label>  
 <input type="text" name="name" th:field="\*{name}">  
 <label>Address</label>  
 <input type="text" name="address" th:field="\*{address}">  
 <label>Address</label>  
 <input type="text" name="address" th:field="\*{email}">  
 <button type="submit">Edit</button>  
</form>  
</body>  
</html>

* **Chạy lại chương trình test chức năng và kiểm tra dưới database**

### **Hướng dẫn nộp bài:**

Up code lên github.

Paste link github vào phần nộp bài.

Link git tham khảo: [nguyendongminhtri/MD4-CRUD-Hibernate (github.com)](https://github.com/nguyendongminhtri/MD4-CRUD-Hibernate)