[Thực hành] Ứng dụng quản lý khách hàng

Mục tiêu

Luyện tập việc sử dụng Spring JPA với Hibernate. Config file hibernate bằng class JAVA. Sử dụng thêm một tầng mới tầng : **repository** để thực hiện viết các câu lệnh Query xuống database

Điều kiện

Có kiến thức căn bản về việc sử dụng Spring JPA với Hibernate.

Mô tả

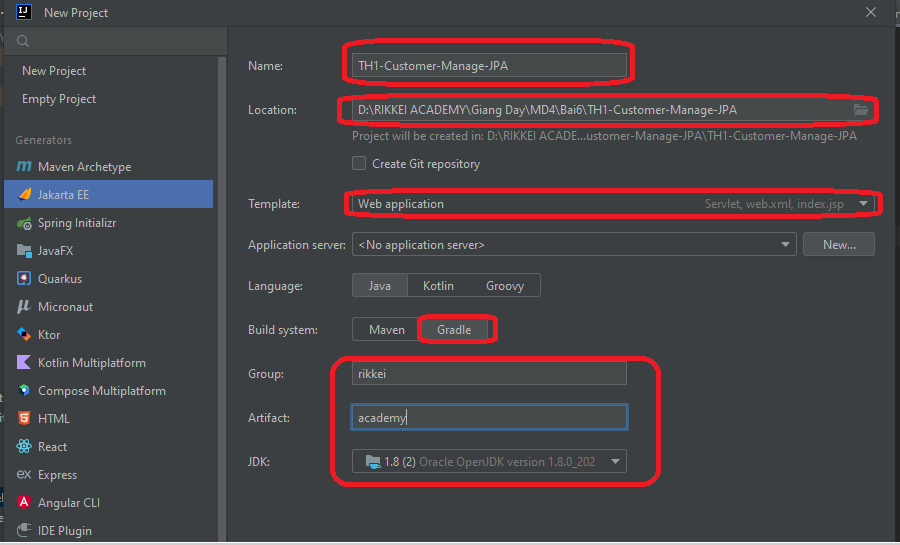
Trong phần này, chúng ta sẽ phát triển một ứng dụng quản lý khách hàng, sử dụng JPA và Hibernate.

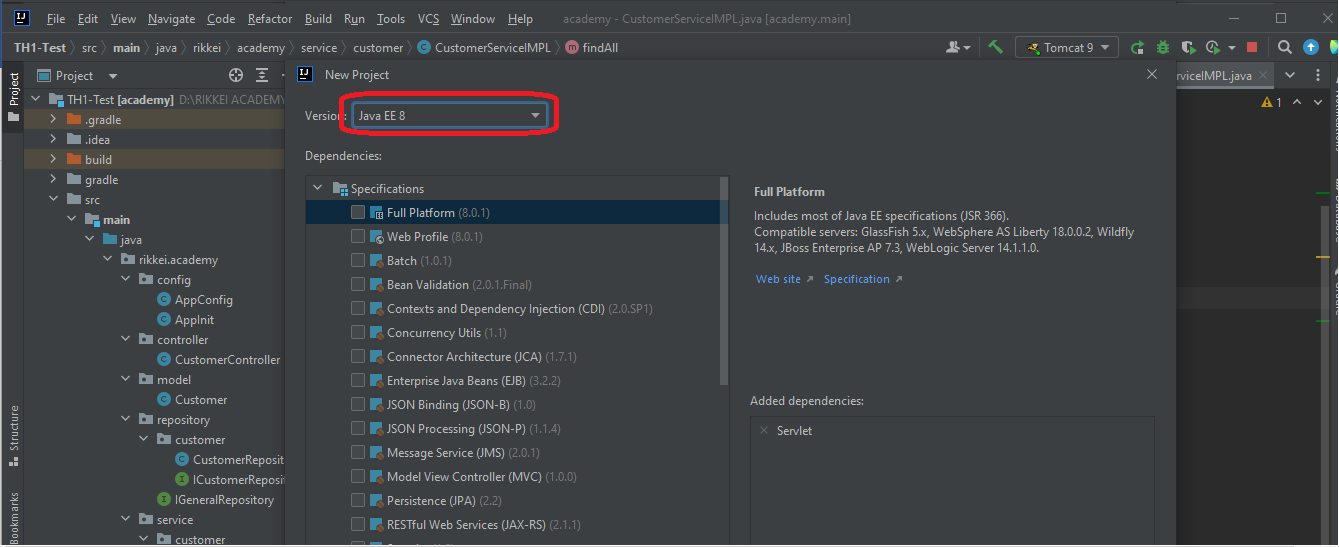
Ứng dụng có các chức năng chính:

* Hiển thị danh sách khách hàng
* Thêm một khách hàng mới
* Xoá một khách hàng
* Chỉnh sửa thông tin khách hàng

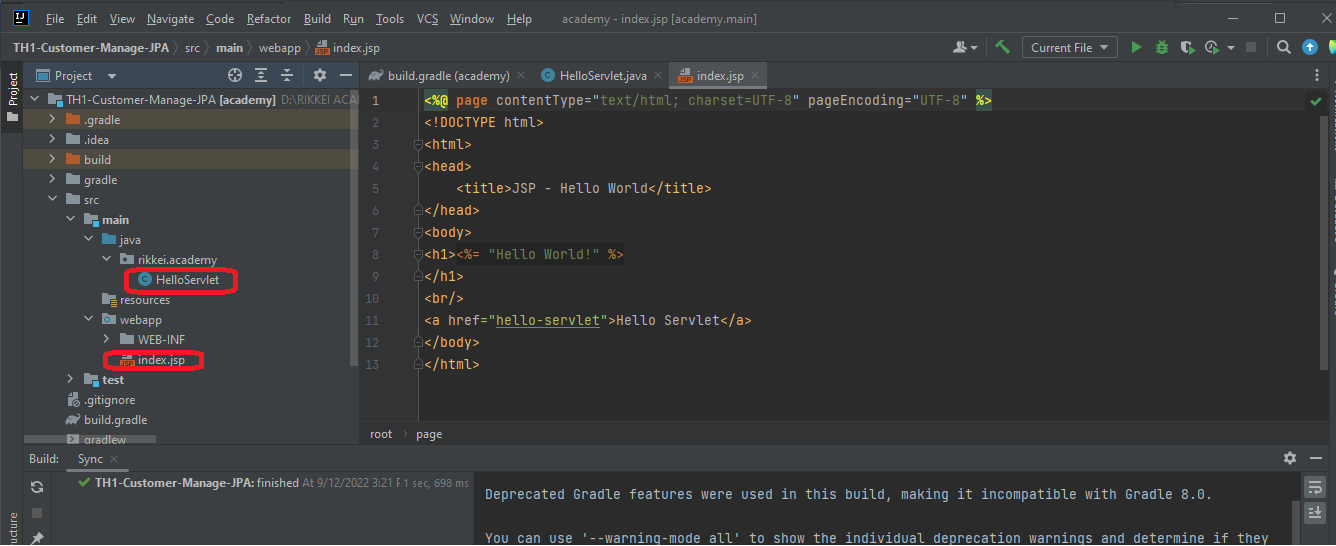
Hướng dẫn:

* Tạo dự án mới:





* Xóa: class: **HelloServlet** và **index.jsp:**



* Tạo package: **config** trong **rikkei.academy**
* Tạo class: **AppInit** trong package: **config** và thêm code vào như sau:

package rikkei.academy.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};  
 }  
}

* Đoạn code:
* @Override  
   protected Filter[] getServletFilters() {  
   CharacterEncodingFilter filter = new CharacterEncodingFilter();  
   filter.setForceEncoding(true);  
   filter.setEncoding("UTF-8");  
   return new Filter[]{filter};  
   }

Để tạo config fonts: unicode UTF-8

* Tạo class: **AppConfig** trong package: **config** và thêm code vào như sau:

package rikkei.academy.config;  
  
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.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  
@ComponentScan("rikkei.academy.controller")  
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("rikkei.academy.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/jpa");  
 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;  
 }  
  
}

* Đoạn code sau tương ứng với file: hibernate.conf.xml trong bài thực hành ở Bài 5 – Các bạn có thể so sánh các chi tiết trong đoạn config này với bài thực hành trên.

@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 package: **model** => Thêm vào class: **Customer** => Thêm code vào như sau:

package rikkei.academy.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() {}  
  
 public Customer(String firstName, String lastName) {  
 this.firstName = firstName;  
 this.lastName = lastName;  
 }  
  
 @Override  
 public String toString() {  
 return String.*format*("Customer[id=%d, firstName='%s', lastName='%s']", id, firstName, lastName);  
 }  
  
 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 package: **repository** => Tạo interface: **IGeneralRepository** => Thêm code vào như sau:

package rikkei.academy.repository;  
  
import java.util.List;  
  
public interface IGeneralRepository<T> {  
 List<T> findAll();  
  
 T findById(Long id);  
  
 void save(T t);  
  
 void remove(Long id);  
}

* Tạo interface: **ICustomerRepository** => Thêm code vào như sau:

package rikkei.academy.repository.customer;  
  
import rikkei.academy.model.Customer;  
import rikkei.academy.repository.IGeneralRepository;  
  
public interface ICustomerRepository extends IGeneralRepository<Customer> {  
}

* Tạo class: **CustomerRepositoryIMPL** => Thêm code vào như sau:

package rikkei.academy.repository.customer;  
  
import org.springframework.transaction.annotation.Transactional;  
import rikkei.academy.model.Customer;  
  
import javax.persistence.EntityManager;  
import javax.persistence.NoResultException;  
import javax.persistence.PersistenceContext;  
import javax.persistence.TypedQuery;  
import java.util.List;  
@Transactional  
public class CustomerRepositoryIMPL implements ICustomerRepository{  
 @PersistenceContext  
 private EntityManager em;  
  
 @Override  
 public List<Customer> findAll() {  
 TypedQuery<Customer> query = em.createQuery("select c from Customer c", Customer.class);  
 return query.getResultList();  
 }  
  
 @Override  
 public Customer findById(Long id) {  
 TypedQuery<Customer> query = em.createQuery("select c from Customer c where c.id=:id", Customer.class);  
 query.setParameter("id", id);  
 try {  
 return query.getSingleResult();  
 } catch (NoResultException e) {  
 return null;  
 }  
 }  
  
 @Override  
 public void save(Customer customer) {  
 if (customer.getId() != null) {  
 em.merge(customer);  
 } else {  
 em.persist(customer);  
 }  
 }  
  
 @Override  
 public void remove(Long id) {  
 Customer customer = findById(id);  
 if (customer != null) {  
 em.remove(customer);  
 }  
 }  
}

* Bổ sung các đoạn code sau vào class: **AppConfig:**

@Bean  
public ICustomerRepository customerRepository() {  
 return new CustomerRepositoryIMPL();  
}

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

package rikkei.academy.service;  
  
import java.util.List;  
  
public interface IGenericService<T> {  
 List<T> findAll();  
  
 T findById(Long id);  
  
 void save(T t);  
  
 void remove(Long id);  
}

* Tạo interface: **ICustomerService** trong package: …**service/customer**

package rikkei.academy.service.customer;  
  
import rikkei.academy.model.Customer;  
import rikkei.academy.service.IGenericService;  
  
public interface ICustomerService extends IGenericService<Customer> {  
}

* Tạo class: **CustomerServiceIMPL** trong package: …**service/customer**

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

* Bổ sung đoạn code sau vào **AppConfig** để tiêm được tầng **service** sang **controller**
* @Bean  
  public ICustomerService customerService() {  
   return new CustomerServiceIMPL();  
  }
* Tạo tầng **controller** => Tạo class: **CustomerController**

package rikkei.academy.controller;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.ModelAttribute;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.servlet.ModelAndView;  
import rikkei.academy.model.Customer;  
import rikkei.academy.service.customer.ICustomerService;  
  
import java.util.List;  
  
@Controller  
public class CustomerController {  
 @Autowired  
 private ICustomerService customerService;  
  
 @GetMapping("/create-customer")  
 public ModelAndView showCreateForm() {  
 ModelAndView modelAndView = new ModelAndView("/customer/create");  
 modelAndView.addObject("customer", new Customer());  
 return modelAndView;  
 }  
  
 @PostMapping("/create-customer")  
 public ModelAndView saveCustomer(@ModelAttribute("customer") Customer customer) {  
 customerService.save(customer);  
 ModelAndView modelAndView = new ModelAndView("/customer/create");  
 modelAndView.addObject("customer", new Customer());  
 modelAndView.addObject("message", "New customer created successfully");  
 return modelAndView;  
 }  
  
 @GetMapping(value = {"/", "/customers"})  
 public ModelAndView listCustomers() {  
 List<Customer> customers = customerService.findAll();  
 ModelAndView modelAndView = new ModelAndView("/customer/list");  
 modelAndView.addObject("customers", customers);  
 return modelAndView;  
 }  
  
 @GetMapping("/edit-customer/{id}")  
 public ModelAndView showEditForm(@PathVariable Long id) {  
 Customer customer = customerService.findById(id);  
 if (customer != null) {  
 ModelAndView modelAndView = new ModelAndView("/customer/edit");  
 modelAndView.addObject("customer", customer);  
 return modelAndView;  
  
 } else {  
 ModelAndView modelAndView = new ModelAndView("/error.404");  
 return modelAndView;  
 }  
 }  
  
 @PostMapping("/edit-customer")  
 public ModelAndView updateCustomer(@ModelAttribute("customer") Customer customer) {  
 customerService.save(customer);  
 ModelAndView modelAndView = new ModelAndView("/customer/edit");  
 modelAndView.addObject("customer", customer);  
 modelAndView.addObject("message", "Customer updated successfully");  
 return modelAndView;  
 }  
  
 @GetMapping("/delete-customer/{id}")  
 public ModelAndView showDeleteForm(@PathVariable Long id) {  
 Customer customer = customerService.findById(id);  
 if (customer != null) {  
 ModelAndView modelAndView = new ModelAndView("/customer/delete");  
 modelAndView.addObject("customer", customer);  
 return modelAndView;  
  
 } else {  
 ModelAndView modelAndView = new ModelAndView("/error.404");  
 return modelAndView;  
 }  
 }  
  
 @PostMapping("/delete-customer")  
 public String deleteCustomer(@ModelAttribute("customer") Customer customer) {  
 customerService.remove(customer.getId());  
 return "redirect:customers";  
 }  
}

* Tạo tầng **views** => Tạo thư mục …**views/customer**
* Tạo file: **list.html**

<!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.thymeleaf.org">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
</head>  
<body>  
<a href="/create-customer">Create new customer</a>  
<h1>Customers</h1>  
<table border="1">  
 <tr>  
 <th>First name</th>  
 <th>Last name</th>  
 <th>Edit</th>  
 <th>Delete</th>  
 </tr>  
 <th:block th:each="customer : ${customers}">  
 <tr>  
 <td th:text="${customer.firstName}"></td>  
 <td th:text="${customer.lastName}"></td>  
 <td><a th:href="@{/edit-customer/\_\_${customer.id}\_\_ }">Edit</a></td>  
 <td><a th:href="@{/delete-customer/\_\_${customer.id}\_\_ }">Delete</a></td>  
 </tr>  
 </th:block>  
</table>  
</body>  
</html>

* Tạo file: **create.html**

<!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.thymeleaf.org">  
<head>  
 <meta charset="UTF-8">  
 <title>Create customer</title>  
</head>  
<body>  
<h1>Create customer</h1>  
<p>  
 <a href="/customers">Customer list</a>  
</p>  
<th:block th:if="${message}">  
 <p th:text="${message}"></p>  
</th:block>  
<form th:action="@{/create-customer}" th:object="${customer}" method="post">  
 <table>  
 <tr>  
 <td>First name:</td>  
 <td><input type="text" th:field="\*{firstName}"/></td>  
 </tr>  
 <tr>  
 <td>Last name:</td>  
 <td><input type="text" th:field="\*{lastName}"/></td>  
 </tr>  
 <tr>  
 <td></td>  
 <td><input type="submit" value="Create customer"></td>  
 </tr>  
 </table>  
</form>  
</body>  
</html>

* Tạo file: **edit.html**

<!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.thymeleaf.org">  
<head>  
 <meta charset="UTF-8">  
 <title>Edit customer</title>  
</head>  
<body>  
<h1>Edit customer</h1>  
<p>  
 <a href="/customers">Customer list</a>  
</p>  
<th:block th:if="${message}">  
 <p th:text="${message}"></p>  
</th:block>  
<form th:action="@{/edit-customer}" th:object="${customer}" method="post">  
 <input th:type="hidden" name="id" th:field="\*{id}">  
 <table>  
 <tr>  
 <td>First name:</td>  
 <td><input type="text" th:field="\*{firstName}"/></td>  
 </tr>  
 <tr>  
 <td>Last name:</td>  
 <td><input type="text" th:field="\*{lastName}"/></td>  
 </tr>  
 <tr>  
 <td></td>  
 <td><input type="submit" value="Update customer"></td>  
 </tr>  
 </table>  
</form>  
</body>  
</html>

* Tạo file: **delete.html**

<!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.thymeleaf.org">  
<head>  
 <meta charset="UTF-8">  
 <title>Delete customer</title>  
</head>  
<body>  
<h1>Delete customer</h1>  
<h2>Are you sure?</h2>  
<p>  
 <a href="/customers">Customer list</a>  
</p>  
<form th:action="@{/delete-customer}" th:object="${customer}" method="post">  
 <input th:type="hidden" name="id" th:field="\*{id}">  
 <table>  
 <tr>  
 <td>First name:</td>  
 <td th:text="${customer.firstName}"></td>  
 </tr>  
 <tr>  
 <td>Last name:</td>  
 <td th:text="${customer.lastName}"></td>  
 </tr>  
 <tr>  
 <td></td>  
 <td><input type="submit" value="Delete customer"></td>  
 </tr>  
 </table>  
</form>  
</body>  
</html>

* Tạo file: **error404.html**

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
</head>  
<body>  
<h1>Error 404 ! Please comeback Menu</h1>  
<p>  
 <a href="/customers">Customer list</a>  
</p>  
</body>  
</html>

* Add: Tomcat và chạy chương trình test các chức năng

Mã nguồn tham khảo:

https://github.com/nguyendongminhtri/MD4-Bai6-TH1-Customer-Manage.git