[Thực hành] Quản lý điện thoại

Mục tiêu

Luyện tập triển khai [AJAX](https://james.codegym.vn/mod/page/view.php?id=2140) trong một ứng dụng web sử dụng jQuery.

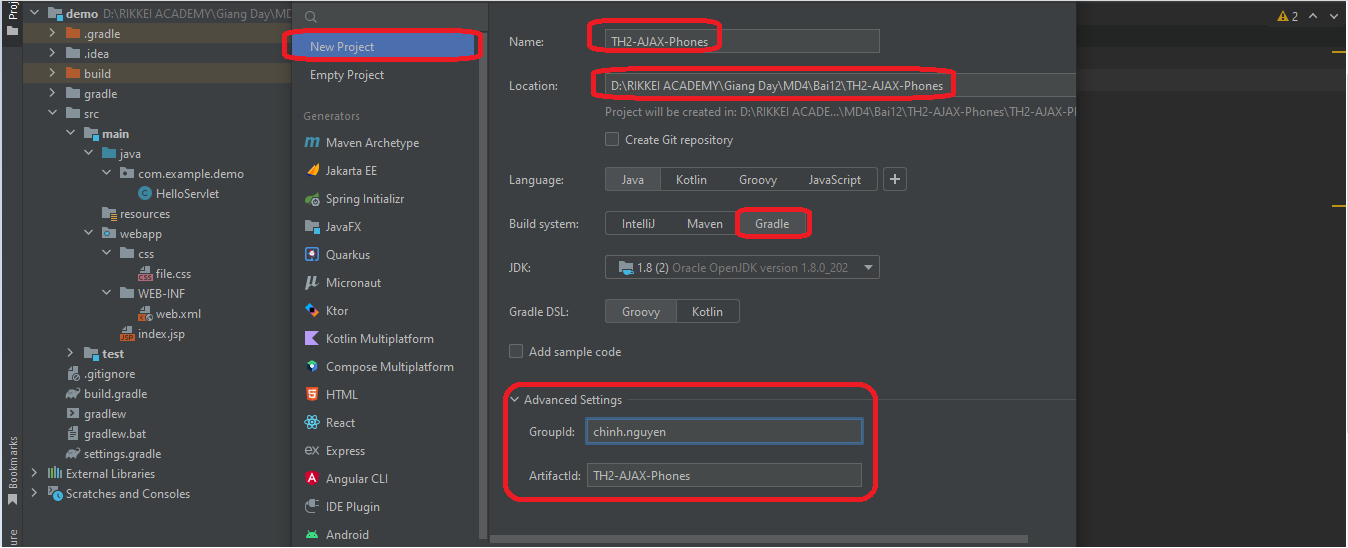
Mô tả

Trong phần này, chúng ta sẽ phát triển một ứng dụng quản lý điện thoại, trong đó có các chức năng:

* Xem danh sách các điện thoại
* Thêm một điện thoại mới
* Chỉnh sửa thông tin điện thoại
* Xoá một điện thoại

Tất các các thao tác của ứng dụng đều được thực hiện dựa trên [AJAX](https://james.codegym.vn/mod/page/view.php?id=2140).

Hướng dẫn



* Import 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'  
 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'  
 implementation group: 'org.springframework', name: 'spring-tx', version: '5.3.2'  
 //Thư viện data json RESTFUL  
// implementation group: 'com.fasterxml.jackson.core', name: 'jackson-databind', version: '2.12.1'  
 implementation group: 'com.fasterxml.jackson.core', name: 'jackson-databind', version: '2.12.1'

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

* Tạo **package**: **chinh.nguyen.config**
* Tạp class: **AppConfig**:

package chinh.nguyen.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.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/bai12\_th\_ajax");  
 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;  
 }  
}

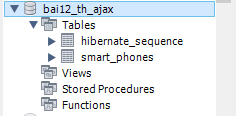
* Tạo class **AppInit**

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 => Tạo class: **SmartPhone:**

package chinh.nguyen.model;  
  
import javax.persistence.\*;  
  
@Entity  
@Table(name="smart\_phones")  
public class SmartPhone {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*AUTO*)  
 private Long id;  
 private String producer;  
 private String model;  
 private double price;  
  
 public SmartPhone() {  
 }  
  
 public SmartPhone(String producer, String model, double price) {  
 this.producer = producer;  
 this.model = model;  
 this.price = price;  
 }  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getProducer() {  
 return producer;  
 }  
  
 public void setProducer(String producer) {  
 this.producer = producer;  
 }  
  
 public String getModel() {  
 return model;  
 }  
  
 public void setModel(String model) {  
 this.model = model;  
 }  
  
 public double getPrice() {  
 return price;  
 }  
  
 public void setPrice(double price) {  
 this.price = price;  
 }  
  
 @Override  
 public String toString() {  
 return producer+": "+model+" with price "+price;  
 }  
  
}

* Add Tomcat và chạy chương trình => Quan sát database trong MySQL



* Tạo package: **repository =>** Tạo interface: **ISmartPhoneRepository**

package chinh.nguyen.repository;  
  
import chinh.nguyen.model.SmartPhone;  
import org.springframework.data.repository.CrudRepository;  
import org.springframework.stereotype.Repository;  
  
@Repository  
public interface ISmartPhoneRepository extends CrudRepository<SmartPhone, Long> {  
 Iterable<SmartPhone> findAllByProducerContaining(String producer);  
}

* Tạo package: **service**
* Tạo interface: **ISmartPhoneService**

package chinh.nguyen.service;  
  
import chinh.nguyen.model.SmartPhone;  
  
import java.util.Optional;  
  
public interface ISmartPhoneService {  
 Iterable<SmartPhone> findAllByProducerContaining(String producer);  
 Iterable<SmartPhone> findAll();  
 Optional<SmartPhone> findById(Long id);  
 void deleteById(Long id);  
 SmartPhone save(SmartPhone smartPhone);  
}

* Tạo class **SmartphoneServiceIMPL**

package chinh.nguyen.service;  
  
import chinh.nguyen.model.SmartPhone;  
import chinh.nguyen.repository.ISmartPhoneRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.Optional;  
@Service  
public class SmartPhoneServiceIMPL implements ISmartPhoneService{  
 @Autowired  
 ISmartPhoneRepository smartPhoneRepository;  
 @Override  
 public Iterable<SmartPhone> findAllByProducerContaining(String producer) {  
 return smartPhoneRepository.findAllByProducerContaining(producer);  
 }  
  
 @Override  
 public Iterable<SmartPhone> findAll() {  
 return smartPhoneRepository.findAll();  
 }  
  
 @Override  
 public Optional<SmartPhone> findById(Long id) {  
 return smartPhoneRepository.findById(id);  
 }  
  
 @Override  
 public void deleteById(Long id) {  
 smartPhoneRepository.deleteById(id);  
 }  
  
 @Override  
 public SmartPhone save(SmartPhone smartPhone) {  
 return smartPhoneRepository.save(smartPhone);  
 }  
}

* Tạo package [**controller**](https://james.codegym.vn/mod/page/view.php?id=1729)**:**  **PhoneController**

package chinh.nguyen.controller;  
  
import chinh.nguyen.model.SmartPhone;  
import chinh.nguyen.service.ISmartPhoneService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
import org.springframework.web.servlet.ModelAndView;  
  
import java.util.Optional;  
  
@RestController  
@RequestMapping("/phone")  
//@CrossOrigin(origins = "\*")  
public class PhoneController {  
 @Autowired  
 ISmartPhoneService smartPhoneService;  
  
 @PostMapping  
 public ResponseEntity<?> createSmartphone(@RequestBody SmartPhone smartphone) {  
 return new ResponseEntity<>(smartPhoneService.save(smartphone), HttpStatus.*CREATED*);  
 }  
  
 @GetMapping  
 public ResponseEntity<?> allPhones() {  
 return new ResponseEntity<>(smartPhoneService.findAll(), HttpStatus.*OK*);  
 }  
  
 @GetMapping("/detail/{id}")  
 public ResponseEntity<?> findById(@PathVariable Long id) {  
 System.*out*.println("goi ham by ID");  
 Optional<SmartPhone> smartPhone = smartPhoneService.findById(id);  
 if (!smartPhone.isPresent()) {  
 return new ResponseEntity<>(HttpStatus.*NOT\_FOUND*);  
 }  
 System.*out*.println("smartPhone"+smartPhone);  
 return new ResponseEntity<>(smartPhone.get(), HttpStatus.*OK*);  
 }  
  
 @DeleteMapping("/delete/{id}")  
 public ResponseEntity<?> deleteSmartphone(@PathVariable Long id) {  
 Optional<SmartPhone> smartphoneOptional = smartPhoneService.findById(id);  
 if (!smartphoneOptional.isPresent()) {  
 return new ResponseEntity<>(HttpStatus.*NOT\_FOUND*);  
 }  
 smartPhoneService.deleteById(id);  
 return new ResponseEntity<>(smartphoneOptional.get(), HttpStatus.*OK*);  
 }  
 @PutMapping("/update/{id}")  
 public ResponseEntity<?> updateSmartPhone(@PathVariable Long id, @RequestBody SmartPhone smartPhone){  
 System.*out*.println("goi ham put");  
 System.*out*.println("producer == "+smartPhone.getProducer());  
 Optional<SmartPhone> smartPhone1 = smartPhoneService.findById(id);  
 if (!smartPhone1.isPresent()) {  
 return new ResponseEntity<>(HttpStatus.*NOT\_FOUND*);  
 }  
 smartPhone1.get().setProducer(smartPhone.getProducer());  
 smartPhone1.get().setModel(smartPhone.getModel());  
 smartPhone1.get().setPrice(smartPhone.getPrice());  
  
 return new ResponseEntity<>(smartPhoneService.save(smartPhone1.get()), HttpStatus.*OK*);  
 }  
}

* Thêm đoạn code vào AppConfig (để chỏ đường dẫn sang các package: controller và repository)

@ComponentScan("chinh.nguyen.controller")  
@EnableJpaRepositories("chinh.nguyen.repository")

* Toàn cảnh AppConfig lúc này:

@Configuration  
@EnableWebMvc  
@EnableTransactionManagement  
@EnableSpringDataWebSupport  
@ComponentScan("chinh.nguyen.controller")  
@EnableJpaRepositories("chinh.nguyen.repository")  
public class AppConfig implements WebMvcConfigurer, ApplicationContextAware {

* Thêm code **@Bean** service vào **AppConfig:**

@Bean  
public ISmartPhoneService smartPhoneService(){  
 return new SmartPhoneServiceIMPL();  
}

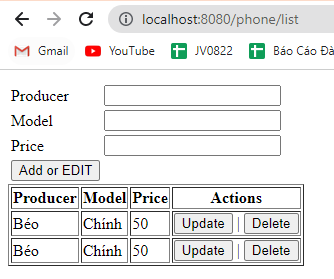
* **Thực hiện test POSTMAN các API - CRUD giống các bài thực hành trước.**
* http://localhost:8080/phone/=> show all (Method Get)
* <http://localhost:8080/phone/detail/1> => detail id 1 (Method Get)
* <http://localhost:8080/phone/update/1> => edit id 1 (Method Put)
* <http://localhost:8080/update/1> => detail id 1 (Method Get)
* Sau khi các test các API kiểm tra thực thi databse nếu các API đều hoạt động tốt ta tạo thêm tầng views: (**webapp/WEB-INF/views**)
* Tạo file: **smart\_phone.html** trong thư mục **views**

<html>  
<head>  
 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>  
 <link rel="stylesheet" type="text/css" href="https://cdn.datatables.net/1.10.18/css/jquery.dataTables.css"/>  
 <script type="text/javascript" src="https://cdn.datatables.net/1.10.18/js/jquery.dataTables.js"></script>  
 <script type = "text/javascript">  
 $(***document***).ready(function () {  
 var book = {};  
 var dynamicURL = "";  
 var methodName = "";  
 getAllBooks();  
 $('#btnAddBook').click(function () {  
 book.producer = $('#producer').val();  
 book.model = $('#model').val();  
 book.price = $('#price').val();  
 var id = $('#id').val();  
 if(id){  
 //update it  
 dynamicURL = "http://localhost:8080/phone/update/"+id;  
 methodName = "PUT";  
 }else{  
 //save it  
 dynamicURL = "http://localhost:8080/phone/";  
 methodName = "POST";  
 }  
 var bookObj = ***JSON***.stringify(book);  
 $.ajax({  
 url: dynamicURL,  
 method: methodName,  
 data: bookObj,  
 contentType: 'application/json; charset=utf-8',  
 success: function () {  
 alert('Saved successfully');  
 getAllBooks();  
 reset();  
 },  
 error: function (error) {  
 alert(error);  
 }  
 })  
 })  
 })  
  
 function getAllBooks() {  
 $.ajax({  
 url: "http://localhost:8080/phone",  
 method: "GET",  
 dataType: "json",  
  
 success: function (data) {  
 var tableBody = $('#tblBook tbody');  
 tableBody.empty();  
 $(data).each(function (index, element) {  
 tableBody.append('<tr><td>'+element.producer+'</td><td>'+element.model+'</td><td>'+element.price+'</td><td><button onclick = "update('+element.id+')">Update</button> | <button onclick = "deleteBook('+element.id+')">Delete</button></td></tr>');  
 })  
 },  
 error: function (error) {  
 alert(error);  
 }  
 })  
 }  
  
 function deleteBook(id){  
 $.ajax({  
 url: 'http://localhost:8080/phone/delete/'+id,  
 method: 'DELETE',  
 success: function () {  
 alert('record has been deleted');  
 getAllBooks();  
 },  
 error: function (error) {  
 alert(error);  
 }  
 })  
 }  
  
 function update(id){  
 ***console***.log('goi ham update', id)  
 $.ajax({  
 url: 'http://localhost:8080/phone/detail/'+id,  
 method: 'GET',  
 dataType: 'json',  
 success: function (data) {  
 ***console***.log("vao success khong?")  
 $('#producer').val(data.producer);  
 $('#model').val(data.model);  
 $('#price').val(data.price);  
 $('#id').val(data.id);  
 getAllBooks();  
 },  
 error: function (error) {  
 alert('loi a'+error);  
 }  
 })  
 }  
  
 function reset(){  
 $('#producer').val('');  
 $('#model').val('');  
 $('#price').val('');  
 $('#id').val('');  
 }  
 </script>  
</head>  
<body>  
<div>  
 <table>  
  
 <td><input type="text" id = "id" hidden></td>  
  
 <tr>  
 <td>Producer</td>  
 <td><input type="text" id = "producer"></td>  
 </tr>  
 <tr>  
 <td>Model</td>  
 <td><input type="text" id = "model"></td>  
 </tr>  
 <tr>  
 <td>Price</td>  
 <td><input type="text" id = "price"></td>  
 </tr>  
 <tr colspan = "2">  
 <td><input type="button" value="Add or EDIT" id="btnAddBook"></td>  
 </tr>  
 </table>  
 <table border = "1" id = "tblBook">  
 <thead>  
 <tr>  
 <th>Producer</th>  
 <th>Model</th>  
 <th>Price</th>  
 <th>Actions</th>  
 </tr>  
 </thead>  
 <tbody>  
  
 </tbody>  
 </table>  
</div>  
</body>  
</html>

* Thêm đoạn code sau vào **PhoneController** để điều hướng sang tầng view:

@GetMapping("/list")  
public ModelAndView getAllSmartphonePage() {  
 ModelAndView modelAndView = new ModelAndView("/smart\_phone");  
 modelAndView.addObject("smartphones", smartPhoneService.findAll());  
 System.*out*.println("modelAndView ---->"+modelAndView);  
 return modelAndView;  
}

* Test với API: <http://localhost:8080/phone/list> => quan sát kết quả:



* Test các chức năng CRUD với giao diện này.
* BÂY GIỜ CHÚNG TA SẼ THỰC HIỆN TÁCH **FrontEnd** và **BackEnd** riêng:
* Thêm code vào **PhoneController**:

@CrossOrigin(origins = "\*")

Dòng code này giúp kết nối 2 cổng khác nhau giữa FrontEnd vs BackEnd. Ví dụ: Cổng BackEnd: <http://localhost:8080> cổng của FE là: http://localhost:4200/

@RestController  
@RequestMapping("/phone")  
@CrossOrigin(origins = "\*")  
public class PhoneController {

//Code PhoneController

}

* Thực hiện chạy lại sever BackEnd.
* Mở phần mềm WEBSTORM tạo file: **index.html**
* Copy phần code ở file: **smart\_phone.html** ở intellij  sang file **index.html** bên webstorm 
* Chạy phần giao diện của Webstorm và test lại các chức năng.

<https://github.com/nguyendongminhtri/MD4-Bai12-AJAX-Smart-Phone.git>

Hướng dẫn nộp bài:

Up code lên github.

Paste link github vào phần nộp bài.

***Hà Nội Ngày Buồn Đêm Lộng Giớ - Chính NĐ -***