**TÀI LIỆU HƯỚNG DẪN LẬP TRÌNH**

**API FRAMEWORK SPRING BOOT**

**BẢNG GHI NHẬN THAY ĐỔI TÀI LIỆU**

| Ngày thay đổi | Người tạo | Lý do | Phiên bản cũ | Mô tả thay đổi | Phiên bản mới |
| --- | --- | --- | --- | --- | --- |
| 29/09/2019 | Hieuntph06393 | Tạo mới |  |  | 1.0 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Mục lục

[**1.** **Giới thiệu** 4](#_Toc23425258)

[**2.** **Cấu trúc Framework** 4](#_Toc23425259)

[**2.1.** **Cấu trúc chung** 4](#_Toc23425260)

[**2.2.** **Cấu trúc module FWApi** 5](#_Toc23425261)

[**2.3.** **Cấu trúc module FWSprBApi** 6](#_Toc23425262)

[**3.** **Cài đặt** 7](#_Toc23425263)

[**4.** **Cơ bản về Spring Boot** 8](#_Toc23425264)

[**5.** **Lập trình với framework** 8](#_Toc23425265)

[**5.1.** **Các quy định chung** 8](#_Toc23425266)

[**5.2.** **Hướng dẫn lập trình chức năng** 9](#_Toc23425267)

[**5.3.** **Hướng dẫn lập trình nâng cao** 12](#_Toc23425268)

[**5.3.1.** **Giao tiếp với hệ thống khác qua API** 12](#_Toc23425269)

[**5.3.2.** **Viết tiến trình chạy ngầm** 13](#_Toc23425270)

[**5.3.3.** **Sử dụng AsyncTask cho các nghiệp vụ chạy bất đồng bộ** 14](#_Toc23425271)

[**6.** **Hướng dẫn build và deploy ứng dụng** 14](#_Toc23425272)

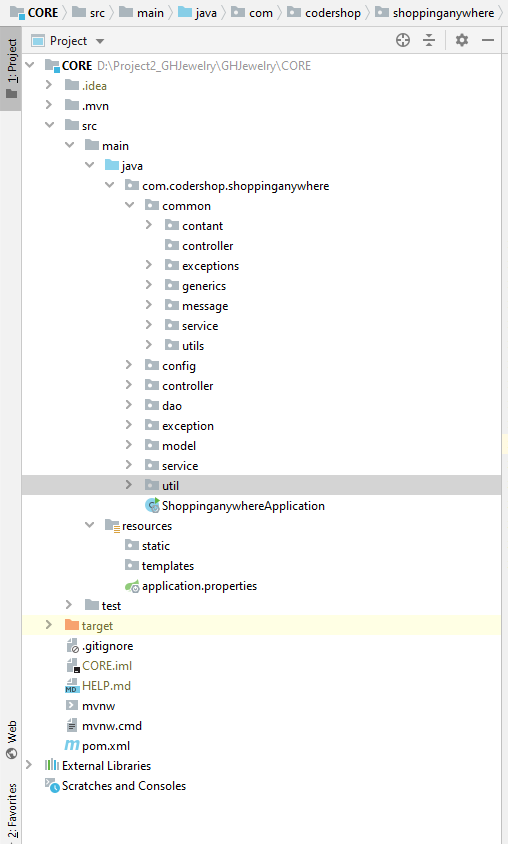
1. **Giới thiệu**

Tài liệu này nhằm mục đích hướng dẫn lập trình phát triển trên nền tảng framework SpringBoot, bao gồm các nội dung chính:

* Quy định khai báo/đặt tên biến, function, class
* Quy định trình bày code
* Hướng dẫn lập trình trên framework đã được xây dựng
* Hướng dẫn build và deploy

1. **Cấu trúc Framework**
   1. **Cấu trúc chung**

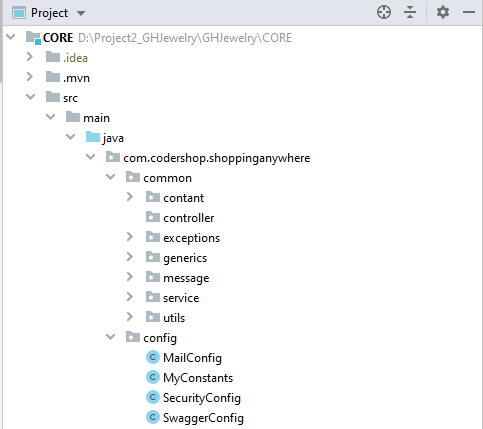
* **Cấu trúc:**



* **Mô tả:**

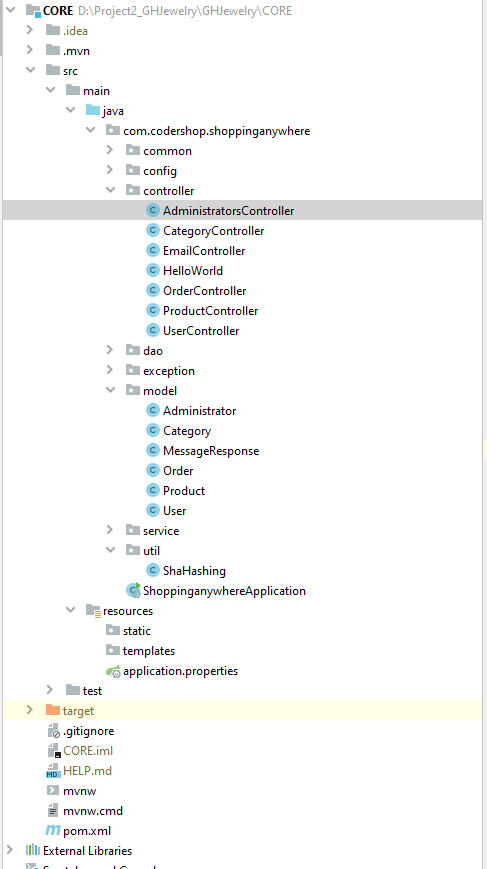
|  |  |  |
| --- | --- | --- |
| **STT** | **File/Thư mục** | **Mô tả** |
| 1 | CORE | Module chứa các mã nguồn sử dụng chung cho toàn bộ module nghiệp vụ. |

* 1. **Cấu trúc module CORE**



|  |  |  |
| --- | --- | --- |
| **STT** | **File/Thư mục** | **Mô tả** |
| 1 | common/exceptions | Chứa các Exception riêng của project |
| 3 | common/generics | Chứa bộ generics Dao, Service cung cấp các xử lý CRUD cơ bản |
| 4 | common/logging | Quản lý log và tiến trình ghi log |
| 5 | common/utils | Chứa các utils dùng chung như StringUtils, DateUtils,.. |
| 6 | config/MailConfig | Xử lý gửi email |
| 7 | config/MyConsants | Chứa các biến dùng chung cho toàn Project |
| 8 | config/SecurityConfig | Chứa các object cho xử lý xác thực truy cập API |
| 9 | config/SwaggerConfig | Hỗ trợ RestAPI |

* 1. **Cấu trúc nghiệp vụ xử lý CORE**



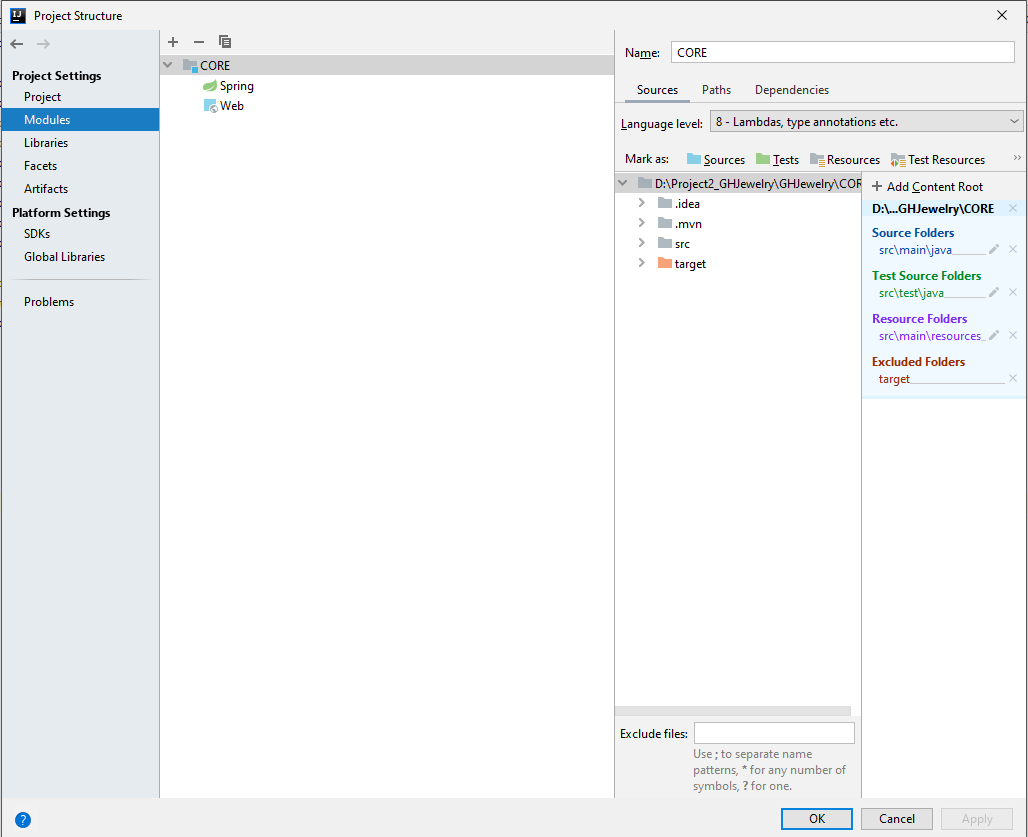
* **Mô tả**:

|  |  |  |
| --- | --- | --- |
| **STT** | **File/Thư mục** | **Mô tả** |
| 1 | Controller | Chứa các Controller public Api |
| 3 | Dao | Chứa các Dao public Api |
| 3 | model | Chứa các entity |
| 4 | ShoppinganywhereApplication | File khởi chạy project |
| 5 | application.properties | Khai báo các cấu hình DB, logging, properties môi trường chung của project |
| 6 | Service | Chứa các Service public Api |
| 7 | Util | Chứa các file |

1. **Cài đặt**

Khi tham gia dự án, khách hàng tạo một source code riêng để tham gia lập trình trên GITHUB, để thực hiện lập trình cần thực hiện các bước sau:

* **Bước 1**: Cài đặt Java 8 (Truy cập trang <https://www.java.com/en/download/> tải bản phù hợp và thực hiện cài đặt)
* **Bước 2**: Checkout source code về máy
* **Bước 3**: Kiếm tra và Add module CORE trong Cấu hình Project



Add dependency trong pom.xml của project mới

<**dependency**>  
 <**groupId**>cmo.cordershop</**groupId**>  
 <**artifactId**>shoppinganywhere</**artifactId**>  
 <**version**>0.0.1-SNAPSHOT</**version**>  
</**dependency**>

* **Bước 5**: Cấu hình port trong application.properties

**server.port**=**8086**

* **Bước 6**: Chạy thử bằng IDE và truy cập địa chỉ sau trên trình duyệt <http://localhost:8086/>
* **Chú ý:** cán bộ lập trình sử dụng công cụ Intellji để lập trình.

1. **Cơ bản về Spring Boot**

Các bạn học tại link: <https://spring.io/guides/gs/spring-boot/>

1. **Lập trình với framework**
   1. **Các quy định chung**

* **Quy tắc đặt tên**

Tên của các thành phần đều viết bằng tiếng anh, có tính gợi nhớ, cụ thể:

* **Tên biến**: là danh từ, chữ cái đầu tiên trong tên viết thường, chữ cái đầu tiên của các từ tiếp theo viết hoa, tên không chứa dấu cách, không chứa ký tự gạch dưới. Ví dụ: btnName, txtOld.
* **Tên class**: là danh từ, tất cả các ký tự đầu tiên của mỗi từ được viết hoa, không chứa dấu cách. Ví dụ: ProductDetail.
* **Tên hàm**: là động từ, chữ cái đầu tiên trong tên viết thường, chữ cái đầu tiên của các từ tiếp theo viết hoa, tên không chứa dấu cách. Ví dụ: getValue, computerData.
* **Tên hằng số**: Toàn bộ các từ được viết hoa, không chứa dấu cách, mỗi từ cách nhau bởi dấu gạch chân. Ví dụ: GET\_PRODUCT.
* **Tên thư mục**: là danh từ, chữ cái đầu tiên trong tên viết thường, chữ cái đầu tiên của các từ tiếp theo viết hoa, tên không chứa dấu cách.
* **Tên file**: Tên file của class giống quy định tên class.
* **Quy tắc về Inject Service vào Controller**:

Mỗi Controller chỉ Autowire 1 Service duy nhất, code business phải implement trong Service, không xử lý trong Controller. Các Service nghiệp vụ khác liên quan sẽ Inject vào Service.

* **Quy tắc ghi log**:

Sử dụng khai báo bằng Slf4j

**import** org.slf4j.Logger;  
**import** org.slf4j.LoggerFactory;

Logger **logger** = LoggerFactory.*getLogger*(getClass());

KHÔNG khai báo bằng các implement khác như Logback, Log4j,…

* 1. **Hướng dẫn lập trình chức năng**

Ví dụ với danh mục đơn giản: Chức năng Cấu hình tham số

* **Bước 1**: Tạo Entity tương ứng với table trong cơ sở dữ liệu
* **package** com.codershop.shoppinganywhere.model;  
    
  **import** com.fasterxml.jackson.annotation.JsonPropertyOrder;  
    
  **import** javax.persistence.\*;  
    
  **import static** javax.persistence.GenerationType.***IDENTITY***;  
    
  @Entity  
  @Table(name = **"users"**, catalog = **"datawebbh"**)  
  @JsonPropertyOrder({**"idUser"**, **"userName"**, **"password"**, **"email"**, **"phone"**, **"address"**})  
  **public class** User {  
   @Id  
   @GeneratedValue(strategy = ***IDENTITY***)  
   @Column(name = **"id\_user"**, unique = **true**, nullable = **false**)  
   **private** Long **idUser**;  
   @Column(name = **"username"**)  
   **private** String **userName**;  
   @Column(name = **"password"**)  
   **private** String **password**;  
   @Column(name = **"email"**)  
   **private** String **email**;  
   @Column(name = **"phone"**)  
   **private** String **phone**;  
   @Column(name = **"address"**)  
   **private** String **address**;  
    
   **public** User(Long idUser, String userName, String password, String email, String phone, String address) {  
   **this**.**idUser** = idUser;  
   **this**.**userName** = userName;  
   **this**.**password** = password;  
   **this**.**email** = email;  
   **this**.**phone** = phone;  
   **this**.**address** = address;  
   }  
    
   **public** User() {  
   }  
    
   **public** Long getIdUser() {  
   **return idUser**;  
   }  
    
   **public void** setIdUser(Long idUser) {  
   **this**.**idUser** = idUser;  
   }  
    
   **public** String getUserName() {  
   **return userName**;  
   }  
    
   **public void** setUserName(String userName) {  
   **this**.**userName** = userName;  
   }  
    
   **public** String getPassword() {  
   **return password**;  
   }  
    
   **public void** setPassword(String password) {  
   **this**.**password** = password;  
   }  
    
   **public** String getEmail() {  
   **return email**;  
   }  
    
   **public void** setEmail(String email) {  
   **this**.**email** = email;  
   }  
    
   **public** String getPhone() {  
   **return phone**;  
   }  
    
   **public void** setPhone(String phone) {  
   **this**.**phone** = phone;  
   }  
    
   **public** String getAddress() {  
   **return address**;  
   }  
    
   **public void** setAddress(String address) {  
   **this**.**address** = address;  
   }  
  }
* **Bước 2**: Tạo repository để sử dụng được các method CRUD

**package** com.cordershop.shoppinganywhere.dao.repo;  
  
**import** org.springframework.data.repository.CrudRepository;  
**import** com.cordershop.shoppinganywhere.model.AppParam;  
  
**interface** UserRepo **extends** CrudRepository<Users,Integer> {}

* **Bước 3**: Tạo Dao và DaoImpl để xử lý với cơ sở dữ liệu

**package** com.cordershop.shoppinganywhere.dao;  
  
  
**import** java.util.List;  
**import** java.util.Map;  
  
**public interface** UserDao **extends** GenericDao<User,Integer> {  
 List<User> getByCodeType(Map map);   
}

**package** com.fis.business.dao.impl;  
  
**import** org.springframework.stereotype.Repository;  
**import** com.fis.fw.common.generics.impl.GenericDaoImpl;  
**import** com.fis.business.dao.AppParamDao;  
**import** com.fis.business.entity.AppParam;  
  
**import** java.util.HashMap;  
**import** java.util.List;  
**import** java.util.Map;  
  
@Repository(**"AppParamDaoImpl"**)  
**public class** AppParamDaoImpl **extends** GenericDaoImpl<AppParam, Integer> **implements** AppParamDao {  
 @Override  
 **public** List<AppParam> getByCodeType(Map map) {  
 String sql = **"select APP\_PARAM\_ID appParamId,CODE code, TYPE type, VALUE value, STATUS status,DESCRIPTION description from ADMIN\_DTHGD.APP\_PARAM where"** +  
 **"(:code is null or CODE = :code)"** +  
 **"and (:type is null or TYPE =:type)"** +  
 **"and (STATUS = 'ACTIVE')"**;  
 **return** getByCondition(map, sql, AppParam.**class**);  
 }  
}

* **Bước 4**: Tạo Service và ServiceImpl để xử lý nghiệp vụ

**package** com.fis.business.service;  
  
**import** com.fis.fw.common.generics.GenericService;  
**import** com.fis.business.entity.AppParam;  
  
**import** java.util.List;  
**import** java.util.Map;  
  
**public interface** AppParamService **extends** GenericService<AppParam, Integer> {  
 List<AppParam> getByCodeType(Map map);   
}

**package** com.fis.business.service.impl;  
  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.beans.factory.annotation.Qualifier;  
**import** org.springframework.stereotype.Service;  
**import** org.springframework.transaction.annotation.Transactional;  
**import** com.fis.fw.common.generics.GenericDao;  
**import** com.fis.fw.common.generics.impl.GenericServiceImpl;  
**import** com.fis.business.dao.AppParamDao;  
**import** com.fis.business.entity.AppParam;  
**import** com.fis.business.service.AppParamService;  
  
**import** java.util.HashMap;  
**import** java.util.List;  
**import** java.util.Map;  
  
@Service(**"AppParamServiceImpl"**)  
@Transactional  
**public class** AppParamServiceImpl **extends** GenericServiceImpl<AppParam, Integer> **implements** AppParamService {  
 @Autowired  
 AppParamDao **appParamDao**;  
  
 **public** AppParamServiceImpl() {  
 }  
  
 @Autowired  
 **public** AppParamServiceImpl(@Qualifier(**"AppParamDaoImpl"**) GenericDao<AppParam, Integer> genericDao) {  
 **super**(genericDao);  
 **this**.**appParamDao** = (AppParamDao) genericDao;  
 }  
  
 @Override  
 **public** List<AppParam> getByCodeType(Map map) {  
 **return appParamDao**.getByCodeType(map);  
 }  
}

* **Bước 5**: Tạo Controller
  + Phần này yêu cầu viết đầy đủ các method cần thiết cho CRUD
  + Các method đều phải khai báo PreAuthorize để xác thực quyền theo PRIVILEGE tương ứng
  + Tất cả method đều phải return ResponseEntity với body là MessagesResponse như đoạn mẫu dưới

**package** com.fis.business.controller;  
  
**import** org.slf4j.Logger;  
**import** org.slf4j.LoggerFactory;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.http.HttpHeaders;  
**import** org.springframework.http.HttpStatus;  
**import** org.springframework.http.ResponseEntity;  
**import** org.springframework.security.access.prepost.PreAuthorize;  
**import** org.springframework.security.core.context.SecurityContextHolder;  
**import** org.springframework.web.bind.annotation.\*;  
**import** com.fis.business.Constant;  
**import** com.fis.business.entity.AppParam;  
**import** com.fis.fw.core.entity.MessagesResponse;  
**import** com.fis.business.service.AppParamService;  
  
**import** java.util.Date;  
**import** java.util.Map;  
  
@RestController  
@CrossOrigin(origins = **"\*"**)  
@RequestMapping(**"/"** + Constant.MENU.***APP\_PARAM***)  
**public class** AppParamController {  
 Logger **logger** = LoggerFactory.*getLogger*(getClass());  
  
 @Autowired  
 AppParamService **appParamService**;  
  
 @RequestMapping(value = {**"/getByCodeType"**}, method = RequestMethod.***POST***)  
 @PreAuthorize(**"@appAuthorizer.authorize(authentication, '"** + Constant.PRIVILEGE.***VIEW*** + **"', this)"**)  
 **public** @ResponseBody ResponseEntity<?> getByCodeType(@RequestBody Map map) {  
 **logger**.info(**" user: "** + SecurityContextHolder.*getContext*().getAuthentication().getName() + Constant.LOG.***BEGIN***);  
 MessagesResponse mess = **new** MessagesResponse();  
 **try** {  
 mess.setData(**appParamService**.getByCodeType(map));  
 **return new** ResponseEntity(mess, HttpStatus.***OK***);  
 } **catch** (Exception e) {  
 **logger**.error(e.getMessage(), e);  
 mess.error(e);  
 **return new** ResponseEntity<>(mess, HttpStatus.***OK***);  
 } **finally** {  
 **logger**.info(**" user: "** + SecurityContextHolder.*getContext*().getAuthentication().getName() + Constant.LOG.***END***);  
 }  
 }

/\* Các method CRUD khác \*/  
}

* 1. **Hướng dẫn lập trình nâng cao**

1. **Hướng dẫn build và deploy ứng dụng**

Trường hợp lỗi ojdbc thì sử dụng lệnh

mvn install:install-file -Dfile=C:\Users\<tên thư mục user>\.m2\repository\com\oracle\ojdbc7\12.1.0.2\ojdbc7-12.1.0.2.jar -DgroupId=com.oracle -DartifactId=ojdbc7 -Dversion=12.1.0.2 -Dpackaging=jar

Build ứng dụng ra file jar

mvn clean package

Copy lên server và chạy