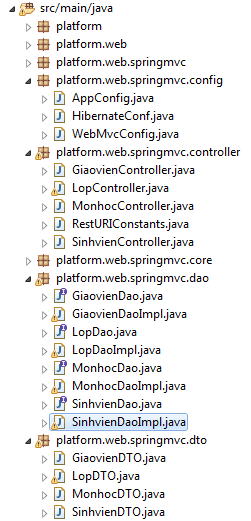
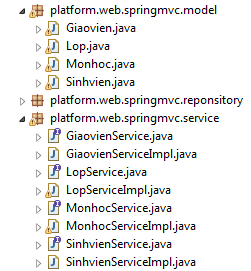
# Spring MVC 5 – Hibernate 5 and MySQL Tutorial

**1. Mô tả bài toán:**

**2. Project Structure:**





**3. Class diagram:**

**4. Maven config:**

<project xmlns=*"http://maven.apache.org/POM/4.0.0"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>Spring</groupId>

<artifactId>SpringMVC\_Hibernate\_MySQL</artifactId>

<packaging>war</packaging>

<version>0.0.1-SNAPSHOT</version>

<name>SpringMVC\_Hibernate\_MySQL Maven Webapp</name>

<url>http://maven.apache.org</url>

<properties>

<org.slf4j-version>1.7.5</org.slf4j-version>

<org.aspectj-version>1.7.4</org.aspectj-version>

<spring.version>5.2.0.RELEASE</spring.version><!-- 5.1.0.RELEASE - 5.2.3.RELEASE -->

<hibernate.version>5.2.11.Final</hibernate.version><!-- 5.3.5.Final - 5.4.14.Final -->

<hibernate.validator>5.4.1.Final</hibernate.validator>

<failOnMissingWebXml>false</failOnMissingWebXml>

<c3p0.version>0.9.5.2</c3p0.version>

<jstl.version>1.2.1</jstl.version>

<tld.version>1.1.2</tld.version>

<servlets.version>3.1.0</servlets.version>

<jsp.version>2.3.1</jsp.version>

<hsqldb.version>1.8.0.10</hsqldb.version>

</properties>

<dependencies>

<!--<dependency> <groupId>tinyWorld.com</groupId> <artifactId>core</artifactId>

<version>1.0</version> </dependency> <dependency> <groupId>tinyWorld.com</groupId>

<artifactId>device-api</artifactId> <version>1.0-SNAPSHOT</version> </dependency>

<dependency> <groupId>org.primefaces</groupId> <artifactId>primefaces</artifactId>

<version>${primefaces-version}</version> </dependency> -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-tx</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

<version>${spring.version}</version>

</dependency>

<!-- Spring Sercurity Config -->

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-config</artifactId>

<version>${spring.version}</version>

</dependency>

<!-- Spring Security Web -->

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-web</artifactId>

<version>${spring.version}</version>

</dependency>

<!-- ====================================================================================

================================ -->

<!-- Dependencies cho viec tao va call rest-service -->

<!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind -->

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

<version>2.10.3</version>

</dependency>

<!-- AspectJ -->

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjrt</artifactId>

<version>${org.aspectj-version}</version>

</dependency>

<!-- Logging -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>${org.slf4j-version}</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>jcl-over-slf4j</artifactId>

<version>${org.slf4j-version}</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<version>${org.slf4j-version}</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.15</version>

<exclusions>

<exclusion>

<groupId>javax.mail</groupId>

<artifactId>mail</artifactId>

</exclusion>

<exclusion>

<groupId>javax.jms</groupId>

<artifactId>jms</artifactId>

</exclusion>

<exclusion>

<groupId>com.sun.jdmk</groupId>

<artifactId>jmxtools</artifactId>

</exclusion>

<exclusion>

<groupId>com.sun.jmx</groupId>

<artifactId>jmxri</artifactId>

</exclusion>

</exclusions>

<scope>runtime</scope>

</dependency>

<!-- @Inject -->

<dependency>

<groupId>javax.inject</groupId>

<artifactId>javax.inject</artifactId>

<version>1</version>

</dependency>

<!-- Servlet -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>servlet-api</artifactId>

<version>2.5</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>javax.servlet.jsp</groupId>

<artifactId>jsp-api</artifactId>

<version>2.1</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

<!-- =====================================================================================

======================= -->

<!-- Depedencies for RestClient -->

<!-- https://mvnrepository.com/artifact/org.codehaus.jackson/jacksonmapper-asl -->

<dependency>

<groupId>org.codehaus.jackson</groupId>

<artifactId>jackson-mapper-asl</artifactId>

<version>1.9.13</version>

</dependency>

<!-- =====================================================================================

======================= -->

<!-- Start: Dependencies for DB Process -->

<!-- https://mvnrepository.com/artifact/org.springframework.data/springdata-jpa -->

<dependency>

<groupId>org.springframework.data</groupId>

<artifactId>spring-data-jpa</artifactId>

<version>2.2.6.RELEASE</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.hibernate/hibernateentitymanager -->

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-entitymanager</artifactId>

<version>${hibernate.version}</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>${hibernate.version}</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-validator</artifactId>

<version>${hibernate.validator}</version>

</dependency>

<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<!-- <version>8.0.11</version> -->

<!-- <version>5.1.9</version> -->

<version>5.1.47</version>

</dependency>

<!-- https://mvnrepository.com/artifact/commons-dbcp/commons-dbcp -->

<dependency>

<groupId>commons-dbcp</groupId>

<artifactId>commons-dbcp</artifactId>

<version>1.4</version>

</dependency>

<dependency>

<groupId>javax.xml.bind</groupId>

<artifactId>jaxb-api</artifactId>

<version>2.3.0</version>

</dependency>

<!-- End Dependencies for DB Process -->

<!-- ============================================================================================================ -->

<!-- Test -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.7</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<finalName>SpringMVC\_Hibernate\_MySQL</finalName>

<sourceDirectory>src/main/java</sourceDirectory>

<resources>

<resource>

<directory>src/main/resources</directory>

</resource>

</resources>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-war-plugin</artifactId>

<version>3.2.2</version>

</plugin>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.5.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

<plugin>

<groupId>org.eclipse.jetty</groupId>

<artifactId>jetty-maven-plugin</artifactId>

<version>9.4.14.v20181114</version>

<configuration> <scanIntervalSeconds>10</scanIntervalSeconds>

<webApp>

<contextPath>/</contextPath>

</webApp>

<httpConnector>

<port>8080</port>

</httpConnector>

</configuration>

</plugin>

<!-- Embedded Apache Tomcat required for testing war -->

<plugin>

<groupId>org.apache.tomcat.maven</groupId>

<artifactId>tomcat-maven-plugin</artifactId>

<version>2.2</version>

<configuration>

<path>/</path>

</configuration>

</plugin>

</plugins>

</build>

</project>

**5. Web.xml:**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns=*"http://java.sun.com/xml/ns/javaee"*

xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee*

*http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"*

id=*"WebApp\_ID"* version=*"3.0"*>

<display-name>Archetype Created Web Application</display-name>

<!-- Context Configuration locations for Spring XML files -->

<welcome-file-list>

<welcome-file>index.jsp</welcome-file>

</welcome-file-list>

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>

classpath:/Beans.xml

</param-value>

</context-param>

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/dispatcher-servlet.xml</param-value>

</context-param>

<!-- <servlet>

<servlet-name>dispatcher</servlet-name>

<servlet-class>

org.springframework.web.servlet.DispatcherServlet

</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>dispatcher</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping> -->

<!-- <listener>

<listener-class>

org.springframework.web.context.ContextLoaderListener

</listener-class>

</listener> -->

</web-app>

**6. DispathcherServlet Configuration:**

Spring cung cấp ***SpringServletContainerInitilalizer*** để xử lí các lớp ***WebApplicationInitializer***. Lớp *AbstractAnnotationConfigDispatcherServletInitializer* thực thi WebMVCConfigurer đây là lớp thực thi *WebApplicationInitializer*. Nó đăng kí ContextLoaderListener (tùy chọn) và một DispatcherServlet cho phép chúng ta dễ dàng thêm vào các configuration classes để nạp các classes cũng như áp dụng các filters vào DispatcherServlet và cung cấp servlet mapping.

Khi sử dụng cấu hình này, ta không cần thêm các đoạn đăng ký servlet vào web.xml như ở các ví dụ trên.

**package** platform.web.springmvc.config;

**import** org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitializer;

/\*@Configuration

@ComponentScan("platform.web.springmvc")\*/

**public** **class** AppConfig **extends** AbstractAnnotationConfigDispatcherServletInitializer {

@Override

**protected** Class<?>[] getRootConfigClasses() {

// **TODO** Auto-generated method stub

**return** **new** Class[] {HibernateConf.**class**};

}

@Override

**protected** Class<?>[] getServletConfigClasses() {

// **TODO** Auto-generated method stub

**return** **new** Class[] {WebMvcConfig.**class**};

}

@Override

**protected** String[] getServletMappings() {

// **TODO** Auto-generated method stub

**return** **new** String[] {"/"};

}

/\*

\* @Bean RestTemplate restTemplate() { RestTemplate restTemplate = new

\* RestTemplate(); MappingJacksonHttpMessageConverter converter = new

\* MappingJacksonHttpMessageConverter(); converter.setObjectMapper(new

\* ObjectMapper()); restTemplate.getMessageConverters().add(converter); return

\* restTemplate; }

\*/

}

**7. Spring WebMVC Configuration:**

**package** platform.web.springmvc.config;

**import** org.springframework.context.MessageSource;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.ComponentScan;

**import** org.springframework.context.annotation.Configuration;

**import** org.springframework.context.support.ResourceBundleMessageSource;

**import** org.springframework.web.servlet.config.annotation.EnableWebMvc;

**import** org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

**import** org.springframework.web.servlet.view.InternalResourceViewResolver;

**import** org.springframework.web.servlet.view.JstlView;

**@Configuration**

**@EnableWebMvc**

**@ComponentScan(basePackages = "platform.web.springmvc")**

**public** **class** WebMvcConfig **implements** WebMvcConfigurer {

**@Bean**

**public** InternalResourceViewResolver resolver() {

**InternalResourceViewResolver** resolver = **new** InternalResourceViewResolver();

resolver.setViewClass(JstlView.**class**);

resolver.setPrefix("/");

resolver.setSuffix(".jsp");

**return** resolver;

}

@Bean

**public** MessageSource messageSource() {

ResourceBundleMessageSource source = **new** ResourceBundleMessageSource();

source.setBasename("messages");

**return** source;

}

/\*

\* public Validator getValidator() { LocalValidatorFactoryBean validator = new

\* LocalValidatorFactoryBean();

\* validator.setValidationMessageSource(messageSource()); return validator; }

\*/

}

Lớp này định nghĩa các option cho việc tùy chỉnh hoặc thêm vào trong default Spring MVC Configuration, nó được kích hoạt thông qua @EnableWebMVC

**@EnableWebMvc**: kích hoạt default Spring MVC Configuration và đăng ký các component hạ tầng mong muốn cho DispathcherServlet.

**@Configuration**: chỉ định một lớp khai báo một hay nhiều @Bean methods và có thể được xử lí bởi Spring Container để tạo các bean definations và service requests cho các bean này tại thời điểm runtime.

**@ComponentScan**: sử dụng để chỉ định base-packages để scan. Bất kỳ class nào được đánh dấu với @Component và @Configuration sẽ được quét.

**InternalResourceViewResolver**: hỗ trợ việc ánh xạ các logical view tới các file cụ thể trong thư mục được chỉ định,

LocalValidatorFactoryBean đưa một javax.validator.ValidationFactory và thể hiện nó trong Spring Validator interface thông qua JSR-303 validator interface và ValidatorFactory interface.

**8. Hibernate Configuration:**

**package** platform.web.springmvc.config;

**import** java.util.Properties;

**import** javax.sql.DataSource;

**import** org.hibernate.SessionFactory;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.ComponentScan;

**import** org.springframework.context.annotation.Configuration;

**import** org.springframework.context.annotation.PropertySource;

**import** org.springframework.core.env.Environment;

**import** org.springframework.jdbc.datasource.DriverManagerDataSource;

**import** org.springframework.orm.hibernate5.HibernateTransactionManager;

**import** org.springframework.orm.hibernate5.LocalSessionFactoryBean;

**import** org.springframework.transaction.annotation.EnableTransactionManagement;

**import** org.springframework.web.servlet.config.annotation.EnableWebMvc;

@Configuration

@EnableWebMvc

@EnableTransactionManagement(proxyTargetClass = **true**)

@PropertySource("classpath:hibernateconfig.properties")

@ComponentScan(basePackages = { "platform.web.springmvc" })

**public** **class** HibernateConf {

// Using environment bean to read property configuration from

// hibernateconfig.properties file

@Autowired

**private** Environment environment;

@Bean

**public** LocalSessionFactoryBean getSessionFactory() {

System.***out***.println("\n\n\n\n>>>>>> HibernateConf.sessionFactory");

LocalSessionFactoryBean sessionFactory = **new** LocalSessionFactoryBean();

sessionFactory.setDataSource(dataSource());

sessionFactory.setPackagesToScan(**new** String[] { "platform.web.springmvc" });

sessionFactory.setHibernateProperties(hibernateProperties());

System.***out***.println(

"-----> HibernateConfiguration--> sessionFactory-->localSesFactory:" + sessionFactory.toString());

/\*

\* System.out.println("-----> HibernateConfiguration--> sessionFactory--

\* >SessonFactory:" + sessionFactory.getObject().toString());

\*/

System.***out***.println(">>>>>>End sessionFactory\n\n\n\n");

**return** sessionFactory;

}

@Bean

**public** DataSource dataSource() {

System.***out***.println(">>>>>> HibernateConf.dataSource");

// BasicDataSource dataSource = new BasicDataSource();

DriverManagerDataSource dataSource = **new** DriverManagerDataSource();

dataSource.setDriverClassName(environment.getRequiredProperty("jdbc.driverClassName"));

dataSource.setUrl(environment.getRequiredProperty("jdbc.url"));

dataSource.setUsername(environment.getRequiredProperty("jdbc.username"));

dataSource.setPassword(environment.getRequiredProperty("jdbc.password"));

System.***out***.println(" - jdbc.driverClassName:" + environment.getRequiredProperty("jdbc.driverClassName"));

System.***out***.println(" - jdbc.url:" + environment.getRequiredProperty("jdbc.url"));

System.***out***.println(" - jdbc.username:" + environment.getRequiredProperty("jdbc.username"));

System.***out***.println(" - jdbc.password: " + environment.getRequiredProperty("jdbc.password"));

System.***out***.println(">>>>>>End dataSource");

**return** dataSource;

}

**private** **final** Properties hibernateProperties() {

Properties properties = **new** Properties();

properties.put("hibernate.dialect", environment.getRequiredProperty("hibernate.dialect"));

properties.put("hibernate.show\_sql", environment.getRequiredProperty("hibernate.show\_sql"));

properties.put("hibernate.format\_sql", environment.getRequiredProperty("hibernate.format\_sql"));

properties.put("hibernate.enable\_lazy\_load\_no\_trans", environment.getRequiredProperty("hibernate.enable\_lazy\_load\_no\_trans"));

properties.put("current\_session\_context\_class", environment.getProperty("current\_session\_context\_class"));

System.***out***.println(" - hibernate.dialect:" + environment.getRequiredProperty("hibernate.dialect"));

System.***out***.println(" - hibernate.show\_sql:" + environment.getRequiredProperty("hibernate.show\_sql"));

System.***out***.println(" - hibernate.format\_sql:" + environment.getRequiredProperty("hibernate.format\_sql"));

//System.out.println(" - hibernate.enable\_lazy\_load\_no\_trans:" + environment.getRequiredProperty("hibernate.enable\_lazy\_load\_no\_trans"));

System.***out***.println(

" - current\_session\_context\_class:" + environment.getProperty("current\_session\_context\_class"));

**return** properties;

}

/\*

\* @Autowired

\*

\* @Bean(name = "transactionManager") public HibernateTransactionManager

\* transactionManager(SessionFactory s) { System.out.println(">>>>>>

\* HibernateConfiguration.transactionManager -- >SessionFactory: " +

\* s.toString()); HibernateTransactionManager txManager = new

\* HibernateTransactionManager(s); // txManager.setSessionFactory(s);

\* System.out.println("\t\t>>>>>> sessionFactory: " + s.toString());

\* System.out.println(">>>>>>End transactionManager\n\n\n\n"); return txManager;

\* }

\*/

@Bean

**public** HibernateTransactionManager getTransactionManager() {

System.***out***.println("\n\n\n\n>>>>>>HibernateConfiguration.getTransactionManager -->SessionFactory: ");

HibernateTransactionManager transactionManager = **new** HibernateTransactionManager();

SessionFactory sessionFactory = getSessionFactory().getObject();

System.***out***.println("\t\t>>>>>> sessionFactory: " + sessionFactory.toString());

transactionManager.setSessionFactory(sessionFactory);

System.***out***.println(">>>>>> End getTransactionManager");

**return** transactionManager;

}

}

* **LocalSessionFactoryBean** tạo ra một Hibernate **SessionFactory**. Đây là cách hữu ích để tạo ra một shared Hibernate SessionFactory trong Spring application context.
* **EnableTransactionManage** kích hoạt annotation-driven transaction manager của Spring.
* **HibernateTransactionManager** ràng buộc một Hibernate Session từ một factory cụ thể vào thread, có khả năng cho phép một thread-bound Session trên một factory. Transaction manager này là phù hợp cho ứng dụng sử dụng một single Hibernate SessionFactory cho transactional data access, nó cũng hỗ trợ direct DataSource truy cập vào trong một transaction i.e. plain JDBC
* **File hibernateconfig.properties**

jdbc.driverClassName = com.mysql.jdbc.Driver

jdbc.url = jdbc:mysql://localhost:3306/my\_project

jdbc.username = root

jdbc.password = 12345678@

hibernate.dialect = org.hibernate.dialect.MySQLDialect

hibernate.show\_sql = true

hibernate.format\_sql = false

hibernate.enable\_lazy\_load\_no\_trans = true;

current\_session\_context\_class = thread

**9. Spring Controller and Restful config:**

* **RestURIConstants**

**package** platform.web.springmvc.controller;

**public** **class** RestURIConstants {

// URI for monhoc

**public** **static** **final** String ***DUMMY\_MONHOC*** = "/rest/monhoc/dummy";

**public** **static** **final** String ***GET\_MONHOC*** = "/rest/monhoc/{id}";

**public** **static** **final** String ***GET\_ALL\_MONHOC*** = "/rest/monhocs";

**public** **static** **final** String ***CREATE\_MONHOC*** = "/rest/monhoc/create";

**public** **static** **final** String ***EDIT\_MONHOC*** = "/rest/monhoc/edit";

**public** **static** **final** String ***DELETE\_MONHOC*** = "/rest/monhoc/{id}";

// URI for lop

**public** **static** **final** String ***DUMMY\_LOP*** = "/rest/lop/dummy";

**public** **static** **final** String ***GET\_LOP*** = "/rest/lop/{id}";

**public** **static** **final** String ***GET\_ALL\_LOP*** = "/rest/lops";

**public** **static** **final** String ***CREATE\_LOP*** = "/rest/lop/create";

**public** **static** **final** String ***EDIT\_LOP*** = "/rest/lop/edit";

**public** **static** **final** String ***DELETE\_LOP*** = "/rest/lop/{id}";

// URI for sinhvien

**public** **static** **final** String ***DUMMY\_SINHVIEN*** = "/rest/sinhvien/dummy";

**public** **static** **final** String ***GET\_SINHVIEN*** = "/rest/sinhvien/{id}";

**public** **static** **final** String ***GET\_ALL\_SINHVIEN*** = "/rest/sinhviens";

**public** **static** **final** String ***CREATE\_SINHVIEN*** = "/rest/sinhvien/create";

**public** **static** **final** String ***EDIT\_SINHVIEN*** = "/rest/sinhvien/edit";

**public** **static** **final** String ***DELETE\_SINHVIEN*** = "/rest/sinhvien/{id}";

// URI for giaovien

**public** **static** **final** String ***DUMMY\_GIAOVIEN*** = "/rest/giaovien/dummy";

**public** **static** **final** String ***GET\_GIAOVIEN*** = "/rest/giaovien/{id}";

**public** **static** **final** String ***GET\_ALL\_GIAOVIEN*** = "/rest/giaoviens";

**public** **static** **final** String ***CREATE\_GIAOVIEN*** = "/rest/giaovien/create";

**public** **static** **final** String ***EDIT\_GIAOVIEN*** = "/rest/giaovien/edit";

**public** **static** **final** String ***DELETE\_GIAOVIEN*** = "/rest/giaovien/{id}";

}

* **MonhocController**

**package** platform.web.springmvc.controller;

**import** java.util.List;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.RequestBody;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** org.springframework.web.bind.annotation.ResponseBody;

**import** platform.web.springmvc.dao.MonhocDao;

**import** platform.web.springmvc.dto.MonhocDTO;

**import** platform.web.springmvc.model.Monhoc;

**import** platform.web.springmvc.service.MonhocService;

@Controller

**public** **class** MonhocController {

@Autowired

**private** MonhocService monhocService;

@Autowired

**private** MonhocDao monhocDao;

**private** **static** **final** Logger ***logger*** = LoggerFactory.*getLogger*(MonhocController.**class**);

@RequestMapping(value = RestURIConstants.***DUMMY\_MONHOC***, method = RequestMethod.***GET***)

**public** @ResponseBody List<Monhoc> getDummyMonHoc() {

***logger***.info("Start getDummyMonHoc");

//MonhocDTO mon = monhocService.getMonhocByID(01);

//System.out.println("\n\n\n\n>>>>>> Monhoc: " + mon.getTenMonhoc());

List<Monhoc> mon = monhocDao.getMonhocByName("java");

**return** mon;

}

@RequestMapping(value = RestURIConstants.***GET\_MONHOC***, method = RequestMethod.***GET***)

**public** @ResponseBody MonhocDTO getMonhoc(@PathVariable("id") **int** id) {

***logger***.info("Start getMonhoc. ID = " + id);

MonhocDTO mon = monhocService.getMonhocByID(id);

**return** mon;

}

@RequestMapping(value = RestURIConstants.***GET\_ALL\_MONHOC***, method = RequestMethod.***GET***)

**public** @ResponseBody List<MonhocDTO> getMonhocs() {

***logger***.info("Start getAllMonhocs");

List<MonhocDTO> list = monhocService.getAll();

**return** list;

}

@RequestMapping(value = RestURIConstants.***CREATE\_MONHOC***, method = RequestMethod.***POST***)

**public** @ResponseBody Monhoc createMonhoc(@RequestBody Monhoc mon) {

***logger***.info("Start createMonhoc");

monhocService.insertMonhoc(mon);

**return** mon;

}

@RequestMapping(value = RestURIConstants.***EDIT\_MONHOC***, method = RequestMethod.***PUT***)

**public** @ResponseBody Monhoc editMonhoc(@RequestBody Monhoc mon) {

***logger***.info("Start editMonhoc. ID = " + mon.getMaMonhoc());

monhocService.editMonhoc(mon);

**return** mon;

}

@RequestMapping(value = RestURIConstants.***DELETE\_MONHOC***, method = RequestMethod.***DELETE***)

**public** @ResponseBody Monhoc deleteMonhoc(@PathVariable("id") **int** id) {

***logger***.info("Start deleteMonhoc. ID = " + id);

monhocService.delete(id);

**return** **null**;

}

}

Trong ví dụ này, ta lưu tất cả dữ liệu vào database

* ***@RequestMapping annotation*** được sử dụng để ánh xạ request URI tới handler method. Ta có thể chỉ định HTTP method được dùng bởi client application để gọi rest method.
* ***@ResponseBody annotation***: được sử dụng để ánh xạ response object trong response body. Mỗi response object được trả về với handler method, MappingJackson2HttpMessageConverter đưa vào và chuyển đổi nó thành JSON response
* ***@PathVariable annotation***: Đây là cacash đơn giản để lấy data từ trong rest URI và ánh xạ nó vào method argument.
* ***@RequestBody annotation:*** Được sử dụng để ánh xạ request body JSON data vào trong Employee object, nó được thực hiện bằng MappingJackson2HttpMessageConverter mapping.
* **LopController**

**package** platform.web.springmvc.controller;

**import** java.util.List;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.RequestBody;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** org.springframework.web.bind.annotation.ResponseBody;

**import** platform.web.springmvc.dto.LopDTO;

**import** platform.web.springmvc.model.Lop;

**import** platform.web.springmvc.service.LopService;

@Controller

**public** **class** LopController {

@Autowired

**private** LopService lopService;

**private** **static** **final** Logger ***logger*** = LoggerFactory.*getLogger*(LopController.**class**);

@RequestMapping(value = RestURIConstants.***DUMMY\_LOP***, method = RequestMethod.***GET***)

**public** @ResponseBody LopDTO getDummyLop() {

***logger***.info("Start getDummyLop");

LopDTO lop = lopService.getLopByID(01);

System.***out***.println("\n\n\n\n>>>>>> Lop: " + lop.getTenLop());

**return** lop;

}

@RequestMapping(value = RestURIConstants.***GET\_LOP***, method = RequestMethod.***GET***)

**public** @ResponseBody LopDTO getLop(@PathVariable("id") **int** id) {

***logger***.info("Start getLop. ID = " + id);

LopDTO lop = lopService.getLopByID(id);

**return** lop;

}

@RequestMapping(value = RestURIConstants.***GET\_ALL\_LOP***, method = RequestMethod.***GET***)

**public** @ResponseBody List<LopDTO> getLops() {

***logger***.info("Start getAllLops");

List<LopDTO> list = lopService.getAll();

**return** list;

}

@RequestMapping(value = RestURIConstants.***CREATE\_LOP***, method = RequestMethod.***POST***)

**public** @ResponseBody Lop createLop(@RequestBody Lop lop) {

***logger***.info("Start createLop");

lopService.insertLop(lop);

**return** lop;

}

@RequestMapping(value = RestURIConstants.***EDIT\_LOP***, method = RequestMethod.***PUT***)

**public** @ResponseBody Lop editLop(@RequestBody Lop lop) {

***logger***.info("Start editeLop. ID = " + lop.getMaLop());

lopService.editLop(lop);

**return** lop;

}

@RequestMapping(value = RestURIConstants.***DELETE\_LOP***, method = RequestMethod.***DELETE***)

**public** @ResponseBody Lop deleteLop(@PathVariable("id") **int** id) {

***logger***.info("Start deleteLop. ID = "+id);

lopService.delete(id);

**return** **null**;

}

}

* **GiaovienController**

**package** platform.web.springmvc.controller;

**import** java.util.List;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.RequestBody;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** org.springframework.web.bind.annotation.ResponseBody;

**import** platform.web.springmvc.dto.GiaovienDTO;

**import** platform.web.springmvc.model.Giaovien;

**import** platform.web.springmvc.service.GiaovienService;

@Controller

**public** **class** GiaovienController {

@Autowired

**private** GiaovienService giaovienService;

**private** **static** **final** Logger ***logger*** = LoggerFactory.*getLogger*(GiaovienController.**class**);

@RequestMapping(value = RestURIConstants.***DUMMY\_GIAOVIEN***, method = RequestMethod.***GET***)

**public** @ResponseBody GiaovienDTO getDummyGiaovien() {

***logger***.info("Start getDummyGiaovien");

GiaovienDTO giaovien = giaovienService.getGiaovienByID(01);

System.***out***.println("\n\n\n\n>>>>>> Giaovien: " + giaovien.getHo() + " " + giaovien.getTen());

**return** giaovien;

}

@RequestMapping(value = RestURIConstants.***GET\_GIAOVIEN***, method = RequestMethod.***GET***)

**public** @ResponseBody GiaovienDTO getGiaovien(@PathVariable("id") **int** id) {

***logger***.info("Start getGiaovien. ID = " + id);

GiaovienDTO giaovien = giaovienService.getGiaovienByID(id);

**return** giaovien;

}

@RequestMapping(value = RestURIConstants.***GET\_ALL\_GIAOVIEN***, method = RequestMethod.***GET***)

**public** @ResponseBody List<GiaovienDTO> getGiaoviens() {

***logger***.info("Start getAllGiaoviens");

List<GiaovienDTO> list = giaovienService.getAll();

**return** list;

}

@RequestMapping(value = RestURIConstants.***CREATE\_GIAOVIEN***, method = RequestMethod.***POST***)

**public** @ResponseBody Giaovien createGiaovien(@RequestBody Giaovien giaovien) {

***logger***.info("Start create Giaovien");

giaovienService.insertGiaovien(giaovien);

**return** giaovien;

}

@RequestMapping(value = RestURIConstants.***EDIT\_GIAOVIEN***, method = RequestMethod.***PUT***)

**public** @ResponseBody Giaovien editGiaovien(@RequestBody Giaovien giaovien) {

***logger***.info("Start editGiaovien. ID = " + giaovien.getMaGiaovien());

giaovienService.editGiaovien(giaovien);

**return** giaovien;

}

@RequestMapping(value = RestURIConstants.***DELETE\_GIAOVIEN***, method = RequestMethod.***DELETE***)

**public** @ResponseBody Giaovien deleteGiaovien(@PathVariable("id") **int** id) {

***logger***.info("Start deleteGiaovien. ID = " + id);

giaovienService.delete(id);

**return** **null**;

}

}

* **SinhvienController**

**package** platform.web.springmvc.controller;

**import** java.util.List;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.RequestBody;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** org.springframework.web.bind.annotation.ResponseBody;

**import** platform.web.springmvc.dto.SinhvienDTO;

**import** platform.web.springmvc.model.Sinhvien;

**import** platform.web.springmvc.service.SinhvienService;

@Controller

**public** **class** SinhvienController {

@Autowired

**private** SinhvienService sinhvienService;

**private** **static** **final** Logger ***logger*** = LoggerFactory.*getLogger*(SinhvienController.**class**);

@RequestMapping(value = RestURIConstants.***DUMMY\_SINHVIEN***, method = RequestMethod.***GET***)

**public** @ResponseBody SinhvienDTO getDummyMonHoc() {

***logger***.info("Start getDummyMonHoc");

SinhvienDTO sinhvien = sinhvienService.getSinhvienByID(01);

System.***out***.println("\n\n\n\n>>>>>> Sinhvien: " + sinhvien.getHo() + " " + sinhvien.getTen());

**return** sinhvien;

}

@RequestMapping(value = RestURIConstants.***GET\_SINHVIEN***, method = RequestMethod.***GET***)

**public** @ResponseBody SinhvienDTO getSinhvien(@PathVariable("id") **int** id) {

***logger***.info("Start getSinhvien. ID = " + id);

SinhvienDTO sinhvien = sinhvienService.getSinhvienByID(id);

**return** sinhvien;

}

@RequestMapping(value = RestURIConstants.***GET\_ALL\_SINHVIEN***, method = RequestMethod.***GET***)

**public** @ResponseBody List<SinhvienDTO> getSinhviens() {

***logger***.info("Start getAllSinhviens");

List<SinhvienDTO> list = sinhvienService.getAll();

**return** list;

}

@RequestMapping(value = RestURIConstants.***CREATE\_SINHVIEN***, method = RequestMethod.***POST***)

**public** @ResponseBody Sinhvien createSinhvien(@RequestBody Sinhvien sinhvien) {

***logger***.info("Start createSinhvien");

sinhvienService.insertSinhvien(sinhvien);

**return** sinhvien;

}

@RequestMapping(value = RestURIConstants.***EDIT\_SINHVIEN***, method = RequestMethod.***PUT***)

**public** @ResponseBody Sinhvien editSinhvien(@RequestBody Sinhvien sinhvien) {

***logger***.info("Start editSinhvien. ID = " + sinhvien.getMaSinhvien());

sinhvienService.editSinhvien(sinhvien);

**return** sinhvien;

}

@RequestMapping(value = RestURIConstants.***DELETE\_SINHVIEN***, method = RequestMethod.***DELETE***)

**public** @ResponseBody Sinhvien deleteSinhvien(@PathVariable("id") **int** id) {

***logger***.info("Start deleteSinhvien. ID = " + id);

sinhvienService.delete(id);

**return** **null**;

}

}

**10. Service layer:**

* **MonhocService**

**package** platform.web.springmvc.service;

**import** java.util.List;

**import** platform.web.springmvc.dto.MonhocDTO;

**import** platform.web.springmvc.model.Monhoc;

**public** **interface** MonhocService {

**public** List<MonhocDTO> getAll();

**public** MonhocDTO getMonhocByID(**int** id);

**public** List<Monhoc> getMonhocByName(String name);

**public** **void** insertMonhoc(Monhoc mon);

**public** **void** editMonhoc(Monhoc mon);

**public** **void** delete(**int** id);

}

* **MonhocServiceImpl**

**package** platform.web.springmvc.service;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** org.springframework.transaction.annotation.Transactional;

**import** platform.web.springmvc.dao.MonhocDao;

**import** platform.web.springmvc.dto.MonhocDTO;

**import** platform.web.springmvc.model.Monhoc;

@Service

**public** **class** MonhocServiceImpl **implements** MonhocService{

@Autowired

MonhocDao monhocDaoImpl;

@Transactional

**public** List<MonhocDTO> getAll() {

**return** monhocDaoImpl.getAll();

}

@Transactional

**public** MonhocDTO getMonhocByID(**int** id) {

**return** monhocDaoImpl.getMonhocByID(id);

}

@Transactional

**public** List<Monhoc> getMonhocByName(String name) {

**return** monhocDaoImpl.getMonhocByName(name);

}

@Transactional

**public** **void** insertMonhoc(Monhoc mon) {

monhocDaoImpl.insertMonhoc(mon);

}

@Transactional

**public** **void** editMonhoc(Monhoc mon) {

monhocDaoImpl.editMonhoc(mon);

}

@Transactional

**public** **void** delete(**int** id) {

monhocDaoImpl.delete(id);

}

}

* ***@Transactional***: đặt ở đầu method hoặc ở đầu class chỉ các method hoặc tất cả các method trong class nằm trong một transaction
* **LopService**

**package** platform.web.springmvc.service;

**import** java.util.List;

**import** platform.web.springmvc.dto.LopDTO;

**import** platform.web.springmvc.model.Lop;

**public** **interface** LopService {

**public** List<LopDTO> getAll();

**public** LopDTO getLopByID(**int** id);

**public** List<Lop> getLopByMonhocID(**int** id);

**public** List<Lop> getLopByName(String name);

**public** **void** insertLop(Lop lop);

**public** **void** editLop(Lop lop);

**public** **void** delete(**int** id);

}

* **LopServiceImpl**

**package** platform.web.springmvc.service;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** org.springframework.transaction.annotation.Transactional;

**import** platform.web.springmvc.dao.LopDao;

**import** platform.web.springmvc.dto.LopDTO;

**import** platform.web.springmvc.model.Lop;

@Service

**public** **class** LopServiceImpl **implements** LopService{

@Autowired

LopDao lopDao;

@Transactional

**public** List<LopDTO> getAll() {

**return** lopDao.getAll();

}

@Transactional

**public** LopDTO getLopByID(**int** id) {

**return** lopDao.getLopByID(id);

}

@Transactional

**public** List<Lop> getLopByMonhocID(**int** id) {

**return** lopDao.getLopByMonhocID(id);

}

@Transactional

**public** List<Lop> getLopByName(String name) {

**return** lopDao.getLopByName(name);

}

@Transactional

**public** **void** insertLop(Lop lop) {

lopDao.insertLop(lop);

}

@Transactional

**public** **void** editLop(Lop lop) {

lopDao.editLop(lop);

}

@Transactional

**public** **void** delete(**int** id) {

lopDao.delete(id);

}

}

* **SinhvienService**

**package** platform.web.springmvc.service;

**import** java.util.List;

**import** platform.web.springmvc.dto.SinhvienDTO;

**import** platform.web.springmvc.model.Sinhvien;

**public** **interface** SinhvienService {

**public** List<SinhvienDTO> getAll();

**public** SinhvienDTO getSinhvienByID(**int** id);

**public** List<Sinhvien> getSinhvienByLopID(**int** id);

**public** List<Sinhvien> getSinhvienByName(String name);

**public** **void** insertSinhvien(Sinhvien sinhvien);

**public** **void** editSinhvien(Sinhvien sinhvien);

**public** **void** delete(**int** id);

}

* **SinhvienServiceImpl**

**package** platform.web.springmvc.service;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** org.springframework.transaction.annotation.Transactional;

**import** platform.web.springmvc.dao.GiaovienDao;

**import** platform.web.springmvc.dto.GiaovienDTO;

**import** platform.web.springmvc.model.Giaovien;

@Service

**public** **class** GiaovienServiceImpl **implements** GiaovienService{

@Autowired

GiaovienDao giaovienDao;

@Transactional

**public** List<GiaovienDTO> getAll() {

**return** giaovienDao.getAll();

}

@Transactional

**public** GiaovienDTO getGiaovienByID(**int** id) {

**return** giaovienDao.getGiaovienByID(id);

}

@Transactional

**public** List<Giaovien> getGiaovienByLopID(**int** id) {

**return** giaovienDao.getGiaovienByLopID(id);

}

@Transactional

**public** List<Giaovien> getGiaovienByName(String name) {

**return** giaovienDao.getGiaovienByName(name);

}

@Transactional

**public** **void** insertGiaovien(Giaovien giaovien) {

giaovienDao.insertGiaovien(giaovien);

}

@Transactional

**public** **void** editGiaovien(Giaovien giaovien) {

giaovienDao.editGiaovien(giaovien);

}

@Transactional

**public** **void** delete(**int** id) {

giaovienDao.delete(id);

}

}

* **GiaovienService**

**package** platform.web.springmvc.service;

**import** java.util.List;

**import** platform.web.springmvc.dto.GiaovienDTO;

**import** platform.web.springmvc.model.Giaovien;

**public** **interface** GiaovienService {

**public** List<GiaovienDTO> getAll();

**public** GiaovienDTO getGiaovienByID(**int** id);

**public** List<Giaovien> getGiaovienByLopID(**int** id);

**public** List<Giaovien> getGiaovienByName(String name);

**public** **void** insertGiaovien(Giaovien giaovien);

**public** **void** editGiaovien(Giaovien giaovien);

**public** **void** delete(**int** id);

}

* **GiaovienServiceImpl**

**package** platform.web.springmvc.service;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** org.springframework.transaction.annotation.Transactional;

**import** platform.web.springmvc.dao.GiaovienDao;

**import** platform.web.springmvc.dto.GiaovienDTO;

**import** platform.web.springmvc.model.Giaovien;

@Service

**public** **class** GiaovienServiceImpl **implements** GiaovienService{

@Autowired

GiaovienDao giaovienDao;

@Transactional

**public** List<GiaovienDTO> getAll() {

**return** giaovienDao.getAll();

}

@Transactional

**public** GiaovienDTO getGiaovienByID(**int** id) {

**return** giaovienDao.getGiaovienByID(id);

}

@Transactional

**public** List<Giaovien> getGiaovienByLopID(**int** id) {

**return** giaovienDao.getGiaovienByLopID(id);

}

@Transactional

**public** List<Giaovien> getGiaovienByName(String name) {

**return** giaovienDao.getGiaovienByName(name);

}

@Transactional

**public** **void** insertGiaovien(Giaovien giaovien) {

giaovienDao.insertGiaovien(giaovien);

}

@Transactional

**public** **void** editGiaovien(Giaovien giaovien) {

giaovienDao.editGiaovien(giaovien);

}

@Transactional

**public** **void** delete(**int** id) {

giaovienDao.delete(id);

}

}

**11. Dao layer:**

* **MonhocDao**

**package** platform.web.springmvc.dao;

**import** java.util.List;

**import** platform.web.springmvc.dto.MonhocDTO;

**import** platform.web.springmvc.model.Monhoc;

**public** **interface** MonhocDao {

**public** List<MonhocDTO> getAll();

**public** MonhocDTO getMonhocByID(**int** id);

**public** List<Monhoc> getMonhocByName(String name);

**public** **void** insertMonhoc(Monhoc mon);

**public** **void** editMonhoc(Monhoc mon);

**public** **void** delete(**int** id);

}

* **MonhocDaoImpl**

**package** platform.web.springmvc.dao;

**import** java.util.List;

**import** org.hibernate.Criteria;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.criterion.Order;

**import** org.hibernate.criterion.Restrictions;

**import** org.hibernate.transform.Transformers;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Repository;

**import** org.springframework.transaction.annotation.Transactional;

**import** platform.web.springmvc.dto.MonhocDTO;

**import** platform.web.springmvc.model.Monhoc;

@Repository

**public** **class** MonhocDaoImpl **implements** MonhocDao {

@Autowired

SessionFactory sessionFactory;

@Transactional

**public** List<MonhocDTO> getAll() {

Session session = sessionFactory.getCurrentSession();

StringBuilder query = **new** StringBuilder();

query.append("Select m.maMonhoc as maMonhoc, m.tenMonhoc as tenMonhoc");

query.append(" from Monhoc m");

List<MonhocDTO> list = session.createQuery(query.toString())

.unwrap(org.hibernate.~~Query~~.**class**)

.~~setResultTransformer~~(Transformers.*aliasToBean*(MonhocDTO.**class**))

.getResultList();

**return** list;

}

@Transactional

**public** MonhocDTO getMonhocByID(**int** id) {

Session session = sessionFactory.getCurrentSession();

Criteria criteria = session.~~createCriteria~~(Monhoc.**class**);

criteria.add(Restrictions.*eq*("id", id));

Monhoc mon = (Monhoc) criteria.list().get(0);

**return** **new** MonhocDTO(mon.getMaMonhoc(), mon.getTenMonhoc());

}

@Transactional

**public** List<Monhoc> getMonhocByName(String name) {

Session session = sessionFactory.getCurrentSession();

Criteria criteria = session.~~createCriteria~~(Monhoc.**class**);

criteria.add(Restrictions.*like*("tenMonhoc", "%" + name + "%"));

criteria.addOrder(Order.*asc*("tenMonhoc"));

List<Monhoc> list = criteria.list();

**return** list;

}

@Transactional

**public** **void** insertMonhoc(Monhoc mon) {

Session session = sessionFactory.getCurrentSession();

session.save(mon);

}

@Transactional

**public** **void** editMonhoc(Monhoc mon) {

Session session = sessionFactory.getCurrentSession();

session.merge(Monhoc.**class**.getSimpleName(), mon);

}

@Transactional

**public** **void** delete(**int** id) {

Session session = sessionFactory.getCurrentSession();

Monhoc mon = (Monhoc) session.get(Monhoc.**class**, id);

**if** (mon != **null**)

session.delete(mon);

}

}

* Những method nằm trong 1 transaction thì chúng ta sẽ gọi ***sessionFactory.getCurrentSession()*** để lấy session chứ không gọi ***sessionFactory.openSession()***
* **LopDao**

**package** platform.web.springmvc.dao;

**import** java.util.List;

**import** platform.web.springmvc.dto.LopDTO;

**import** platform.web.springmvc.model.Lop;

**public** **interface** LopDao {

**public** List<LopDTO> getAll();

**public** LopDTO getLopByID(**int** id);

**public** List<Lop> getLopByMonhocID(**int** id);

**public** List<Lop> getLopByName(String name);

**public** **void** insertLop(Lop lop);

**public** **void** editLop(Lop lop);

**public** **void** delete(**int** id);

}

* **LopDaoImpl**

**package** platform.web.springmvc.dao;

**import** java.util.ArrayList;

**import** java.util.List;

**import** org.hibernate.Criteria;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.criterion.Order;

**import** org.hibernate.criterion.Restrictions;

**import** org.hibernate.transform.Transformers;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Repository;

**import** org.springframework.transaction.annotation.Transactional;

**import** platform.web.springmvc.dto.LopDTO;

**import** platform.web.springmvc.model.Lop;

@Repository

**public** **class** LopDaoImpl **implements** LopDao {

@Autowired

SessionFactory sessionFactory;

@Transactional

**public** List<LopDTO> getAll() {

Session session = sessionFactory.getCurrentSession();

StringBuilder query = **new** StringBuilder();

query.append("Select l.maLop as maLop, l.tenLop as tenLop, l.ngayBatdau as ngayBatdau,");

query.append(" l.ngayKetthuc as ngayKetthuc, l.soGioHoc as soGioHoc");

query.append(" from Lop as l");

List<LopDTO> list = session.createQuery(query.toString()).unwrap(org.hibernate.~~Query~~.**class**)

.~~setResultTransformer~~(Transformers.*aliasToBean*(LopDTO.**class**)).getResultList();

**return** list;

}

@Transactional

**public** LopDTO getLopByID(**int** id) {

Session session = sessionFactory.getCurrentSession();

Criteria criteria = session.~~createCriteria~~(Lop.**class**);

criteria.add(Restrictions.*eq*("id", id));

Lop l = (Lop) criteria.list().get(0);

**return** **new** LopDTO(l.getMaLop(), l.getTenLop(), l.getNgayBatdau(), l.getNgayKetthuc(), l.getSoGioHoc());

}

@Transactional

**public** List<Lop> getLopByName(String name) {

Session session = sessionFactory.getCurrentSession();

Criteria criteria = session.~~createCriteria~~(Lop.**class**);

criteria.add(Restrictions.*like*("tenLop", "%" + name + "%"));

criteria.addOrder(Order.*asc*("tenLop"));

List<Lop> list = criteria.list();

**return** list;

}

@Transactional

**public** **void** insertLop(Lop lop) {

Session session = sessionFactory.getCurrentSession();

session.save(lop);

}

@Transactional

**public** **void** editLop(Lop lop) {

Session session = sessionFactory.getCurrentSession();

session.merge(Lop.**class**.getSimpleName(), lop);

}

@Transactional

**public** **void** delete(**int** id) {

Session session = sessionFactory.getCurrentSession();

Lop lop = (Lop) session.get(Lop.**class**, id);

**if** (lop != **null**)

session.delete(lop);

}

@Transactional

**public** List<Lop> getLopByMonhocID(**int** id) {

Session session = sessionFactory.getCurrentSession();

List<Lop> list = **new** ArrayList<Lop>();

list.addAll(session.createQuery("From Lop l Where l.monhoc.maMonhoc=:monID").setParameter("monID", id).list());

**return** list;

}

}

* **SinhvienDao**

**package** platform.web.springmvc.dao;

**import** java.util.List;

**import** platform.web.springmvc.dto.SinhvienDTO;

**import** platform.web.springmvc.model.Sinhvien;

**public** **interface** SinhvienDao {

**public** List<SinhvienDTO> getAll();

**public** SinhvienDTO getSinhvienByID(**int** id);

**public** List<Sinhvien> getSinhvienByLopID(**int** id);

**public** List<Sinhvien> getSinhvienByName(String name);

**public** **void** insertSinhvien(Sinhvien sinhvien);

**public** **void** editSinhvien(Sinhvien sinhvien);

**public** **void** delete(**int** id);

}

* **SinhvienDaoImpl**

**package** platform.web.springmvc.dao;

**import** java.util.ArrayList;

**import** java.util.List;

**import** org.hibernate.Criteria;

**import** org.hibernate.~~Query~~;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.criterion.Order;

**import** org.hibernate.criterion.Restrictions;

**import** org.hibernate.transform.Transformers;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Repository;

**import** org.springframework.transaction.annotation.Transactional;

**import** platform.web.springmvc.dto.SinhvienDTO;

**import** platform.web.springmvc.model.Sinhvien;

@Repository

**public** **class** SinhvienDaoImpl **implements** SinhvienDao {

@Autowired

SessionFactory sessionFactory;

@Transactional

**public** List<SinhvienDTO> getAll() {

Session session = sessionFactory.getCurrentSession();

StringBuilder query = **new** StringBuilder();

query.append("Select sv.maSinhvien as maSinhvien, sv.ho as ho, sv.ten as ten");

query.append(" from Sinhvien sv");

List<SinhvienDTO> list = session.createQuery(query.toString())

.unwrap(~~Query~~.**class**)

.~~setResultTransformer~~(Transformers.*aliasToBean*(SinhvienDTO.**class**))

.getResultList();

**return** list;

}

@Transactional

**public** SinhvienDTO getSinhvienByID(**int** id) {

Session session = sessionFactory.getCurrentSession();

Criteria criteria = session.~~createCriteria~~(Sinhvien.**class**);

criteria.add(Restrictions.*eq*("maSinhvien", id));

Sinhvien sv = (Sinhvien) criteria.list().get(0);

**return** **new** SinhvienDTO(sv.getMaSinhvien(), sv.getHo(), sv.getTen());

}

@Transactional

**public** List<Sinhvien> getSinhvienByLopID(**int** id) {

Session session = sessionFactory.getCurrentSession();

List<Sinhvien> list = **new** ArrayList<Sinhvien>();

list.add((Sinhvien) session.createQuery("From Sinhvien sv where sv.lop.maLop =:lopID").setParameter("lopID", id)

.list());

**return** list;

}

@Transactional

**public** List<Sinhvien> getSinhvienByName(String name) {

Session session = sessionFactory.getCurrentSession();

Criteria criteria = session.~~createCriteria~~(Sinhvien.**class**);

criteria.add(Restrictions.*like*("tenSinhvien", "%" + name + "%"));

criteria.addOrder(Order.*asc*("tenSinhvien"));

List<Sinhvien> list = criteria.list();

**return** list;

}

@Transactional

**public** **void** insertSinhvien(Sinhvien sinhvien) {

Session session = sessionFactory.getCurrentSession();

session.save(sinhvien);

}

@Transactional

**public** **void** editSinhvien(Sinhvien sinhvien) {

Session session = sessionFactory.getCurrentSession();

session.merge(Sinhvien.**class**.getSimpleName(), sinhvien);

}

@Transactional

**public** **void** delete(**int** id) {

Session session = sessionFactory.getCurrentSession();

Sinhvien sinhvien = (Sinhvien) session.get(Sinhvien.**class**, id);

**if** (sinhvien != **null**)

session.delete(sinhvien);

}

}

**12. Model layer:**

* **Monhoc**

**package** platform.web.springmvc.model;

**import** java.util.HashSet;

**import** java.util.Set;

**import** javax.persistence.CascadeType;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.FetchType;

**import** javax.persistence.GeneratedValue;

**import** **static** javax.persistence.GenerationType.***IDENTITY***;

**import** javax.persistence.Id;

**import** javax.persistence.OneToMany;

**import** javax.persistence.Table;

@Entity

@Table(name = "monhoc", catalog = "my\_project")

**public** **class** Monhoc **implements** java.io.Serializable {

**private** Integer maMonhoc;

**private** String tenMonhoc;

**private** Set<Lop> lops = **new** HashSet<Lop>(0);

**public** Monhoc() {

}

**public** Monhoc(Integer maMon) {

**this**.maMonhoc = maMon;

}

**public** Monhoc(Integer maMon,String tenMonhoc) {

**this**.maMonhoc = maMon;

**this**.tenMonhoc = tenMonhoc;

}

**public** Monhoc(String tenMonhoc) {

**this**.tenMonhoc = tenMonhoc;

}

**public** Monhoc(String tenMonhoc, Set<Lop> lops) {

**this**.tenMonhoc = tenMonhoc;

**this**.lops = lops;

}

@Id

@GeneratedValue(strategy = ***IDENTITY***)

@Column(name = "ma\_monhoc", unique = **true**, nullable = **false**)

**public** Integer getMaMonhoc() {

**return** **this**.maMonhoc;

}

**public** **void** setMaMonhoc(Integer maMonhoc) {

**this**.maMonhoc = maMonhoc;

}

@Column(name = "ten\_monhoc", nullable = **false**, length = 45)

**public** String getTenMonhoc() {

**return** **this**.tenMonhoc;

}

**public** **void** setTenMonhoc(String tenMonhoc) {

**this**.tenMonhoc = tenMonhoc;

}

@OneToMany(fetch = FetchType.***LAZY***, mappedBy = "monhoc",cascade=CascadeType.***ALL***)

**public** Set<Lop> getLops() {

**return** **this**.lops;

}

**public** **void** setLops(Set<Lop> lops) {

**this**.lops = lops;

}

@Override

**public** String toString() {

**return** "Monhoc [maMonhoc=" + maMonhoc + ", tenMonhoc=" + tenMonhoc + "]";

}

}

* **Lop**

**package** platform.web.springmvc.model;

**import** java.util.Date;

**import** java.util.HashSet;

**import** java.util.Set;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.FetchType;

**import** javax.persistence.GeneratedValue;

**import** **static** javax.persistence.GenerationType.***IDENTITY***;

**import** javax.persistence.Id;

**import** javax.persistence.JoinColumn;

**import** javax.persistence.JoinTable;

**import** javax.persistence.ManyToMany;

**import** javax.persistence.ManyToOne;

**import** javax.persistence.Table;

**import** javax.persistence.Temporal;

**import** javax.persistence.TemporalType;

**import** org.springframework.format.annotation.DateTimeFormat;

@Entity

@Table(name = "lop", catalog = "my\_project")

**public** **class** Lop **implements** java.io.Serializable {

**private** Integer maLop;

**private** Monhoc monhoc;

**private** String tenLop;

@DateTimeFormat(pattern = "yyyy-MM-dd")

// or use @DateTimeFormat(pattern = DateTimeFormat.ISO.DATE)

**private** Date ngayBatdau;

@DateTimeFormat(pattern = "yyyy-MM-dd")

**private** Date ngayKetthuc;

**private** **int** soGioHoc;

**private** Set<Giaovien> giaoviens = **new** HashSet<Giaovien>(0);

**private** Set<Sinhvien> sinhviens = **new** HashSet<Sinhvien>(0);

**public** Lop() {

}

**public** Lop(Monhoc monhoc, String tenLop, Date ngayBatdau, Date ngayKetthuc, **int** soGioHoc) {

**this**.monhoc = monhoc;

**this**.tenLop = tenLop;

**this**.ngayBatdau = ngayBatdau;

**this**.ngayKetthuc = ngayKetthuc;

**this**.soGioHoc = soGioHoc;

}

**public** Lop(Monhoc monhoc, String tenLop, Date ngayBatdau, Date ngayKetthuc, **int** soGioHoc, Set<Giaovien> giaoviens,

Set<Sinhvien> sinhviens) {

**this**.monhoc = monhoc;

**this**.tenLop = tenLop;

**this**.ngayBatdau = ngayBatdau;

**this**.ngayKetthuc = ngayKetthuc;

**this**.soGioHoc = soGioHoc;

**this**.giaoviens = giaoviens;

**this**.sinhviens = sinhviens;

}

@Id

@GeneratedValue(strategy = ***IDENTITY***)

@Column(name = "ma\_lop", unique = **true**, nullable = **false**)

**public** Integer getMaLop() {

**return** **this**.maLop;

}

**public** **void** setMaLop(Integer maLop) {

**this**.maLop = maLop;

}

@ManyToOne(fetch = FetchType.***EAGER***)

@JoinColumn(name = "ma\_monhoc", nullable = **false**)

**public** Monhoc getMonhoc() {

**return** **this**.monhoc;

}

**public** **void** setMonhoc(Monhoc monhoc) {

**this**.monhoc = monhoc;

}

@Column(name = "ten\_lop", nullable = **false**, length = 45)

**public** String getTenLop() {

**return** **this**.tenLop;

}

**public** **void** setTenLop(String tenLop) {

**this**.tenLop = tenLop;

}

@Temporal(TemporalType.***DATE***)

@Column(name = "ngay\_batdau", nullable = **false**, length = 10)

**public** Date getNgayBatdau() {

**return** **this**.ngayBatdau;

}

**public** **void** setNgayBatdau(Date ngayBatdau) {

**this**.ngayBatdau = ngayBatdau;

}

@Temporal(TemporalType.***DATE***)

@Column(name = "ngay\_ketthuc", nullable = **false**, length = 10)

**public** Date getNgayKetthuc() {

**return** **this**.ngayKetthuc;

}

**public** **void** setNgayKetthuc(Date ngayKetthuc) {

**this**.ngayKetthuc = ngayKetthuc;

}

@Column(name = "so\_gio\_hoc", nullable = **false**)

**public** **int** getSoGioHoc() {

**return** **this**.soGioHoc;

}

**public** **void** setSoGioHoc(**int** soGioHoc) {

**this**.soGioHoc = soGioHoc;

}

@ManyToMany(fetch = FetchType.***LAZY***)

@JoinTable(name = "giaovien\_lop", joinColumns = { @JoinColumn(name = "ma\_lop") }, inverseJoinColumns = {

@JoinColumn(name = "ma\_giaovien") })

**public** Set<Giaovien> getGiaoviens() {

**return** **this**.giaoviens;

}

**public** **void** setGiaoviens(Set<Giaovien> giaoviens) {

**this**.giaoviens = giaoviens;

}

@ManyToMany(fetch = FetchType.***LAZY***)

@JoinTable(name = "sinhvien\_lop", joinColumns = { @JoinColumn(name = "ma\_lop") }, inverseJoinColumns = {

@JoinColumn(name = "ma\_sinhvien") })

**public** Set<Sinhvien> getSinhviens() {

**return** **this**.sinhviens;

}

**public** **void** setSinhviens(Set<Sinhvien> sinhviens) {

**this**.sinhviens = sinhviens;

}

@Override

**public** String toString() {

**return** "Lop [maLop=" + maLop + ", tenLop=" + tenLop + "]";

}

}

* **Sinhvien**

**package** platform.web.springmvc.model;

**import** java.util.HashSet;

**import** java.util.Set;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.FetchType;

**import** javax.persistence.Id;

**import** javax.persistence.JoinColumn;

**import** javax.persistence.JoinTable;

**import** javax.persistence.ManyToMany;

**import** javax.persistence.Table;

@Entity

@Table(name = "sinhvien", catalog = "my\_project")

**public** **class** Sinhvien **implements** java.io.Serializable {

**private** **int** maSinhvien;

**private** String ho;

**private** String ten;

**private** Set<Lop> lops = **new** HashSet<Lop>(0);

**public** Sinhvien() {

}

**public** Sinhvien(**int** maSinhvien, String ho, String ten) {

**this**.maSinhvien = maSinhvien;

**this**.ho = ho;

**this**.ten = ten;

}

**public** Sinhvien(**int** maSinhvien, String ho, String ten, Set<Lop> lops) {

**this**.maSinhvien = maSinhvien;

**this**.ho = ho;

**this**.ten = ten;

**this**.lops = lops;

}

@Id

@Column(name = "ma\_sinhvien", unique = **true**, nullable = **false**)

**public** **int** getMaSinhvien() {

**return** **this**.maSinhvien;

}

**public** **void** setMaSinhvien(**int** maSinhvien) {

**this**.maSinhvien = maSinhvien;

}

@Column(name = "ho", nullable = **false**, length = 20)

**public** String getHo() {

**return** **this**.ho;

}

**public** **void** setHo(String ho) {

**this**.ho = ho;

}

@Column(name = "ten", nullable = **false**, length = 15)

**public** String getTen() {

**return** **this**.ten;

}

**public** **void** setTen(String ten) {

**this**.ten = ten;

}

@ManyToMany(fetch = FetchType.***LAZY***)

@JoinTable(name = "sinhvien\_lop", catalog = "my\_project", joinColumns = {

@JoinColumn(name = "ma\_sinhvien", nullable = **false**, updatable = **false**) }, inverseJoinColumns = {

@JoinColumn(name = "ma\_lop", nullable = **false**, updatable = **false**) })

**public** Set<Lop> getLops() {

**return** **this**.lops;

}

**public** **void** setLops(Set<Lop> lops) {

**this**.lops = lops;

}

@Override

**public** String toString() {

**return** "Sinhvien [maSinhvien=" + maSinhvien + ", ho=" + ho + ", ten=" + ten + ", lops=" + lops + "]";

}

}

* **Giaovien**

**package** platform.web.springmvc.model;

**import** java.util.HashSet;

**import** java.util.Set;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.FetchType;

**import** javax.persistence.GeneratedValue;

**import** **static** javax.persistence.GenerationType.***IDENTITY***;

**import** javax.persistence.Id;

**import** javax.persistence.JoinColumn;

**import** javax.persistence.JoinTable;

**import** javax.persistence.ManyToMany;

**import** javax.persistence.Table;

@Entity

@Table(name = "giaovien", catalog = "my\_project")

**public** **class** Giaovien **implements** java.io.Serializable {

**private** Integer maGiaovien;

**private** String ho;

**private** String ten;

**private** Set<Lop> lops = **new** HashSet<Lop>(0);

**public** Giaovien() {

}

**public** Giaovien(Integer maGiaovien) {

**this**.maGiaovien = maGiaovien;

}

**public** Giaovien(String ho, String ten) {

**this**.ho = ho;

**this**.ten = ten;

}

**public** Giaovien(String ho, String ten, Set<Lop> lops) {

**this**.ho = ho;

**this**.ten = ten;

**this**.lops = lops;

}

@Id

@GeneratedValue(strategy = ***IDENTITY***)

@Column(name = "ma\_giaovien", unique = **true**, nullable = **false**)

**public** Integer getMaGiaovien() {

**return** **this**.maGiaovien;

}

**public** **void** setMaGiaovien(Integer maGiaovien) {

**this**.maGiaovien = maGiaovien;

}

@Column(name = "ho", nullable = **false**, length = 20)

**public** String getHo() {

**return** **this**.ho;

}

**public** **void** setHo(String ho) {

**this**.ho = ho;

}

@Column(name = "ten", nullable = **false**, length = 15)

**public** String getTen() {

**return** **this**.ten;

}

**public** **void** setTen(String ten) {

**this**.ten = ten;

}

@ManyToMany(fetch = FetchType.***LAZY***)

@JoinTable(name = "giaovien\_lop", catalog = "my\_project", joinColumns = {

@JoinColumn(name = "ma\_giaovien", nullable = **false**, updatable = **false**) }, inverseJoinColumns = {

@JoinColumn(name = "ma\_lop", nullable = **false**, updatable = **false**) })

**public** Set<Lop> getLops() {

**return** **this**.lops;

}

**public** **void** setLops(Set<Lop> lops) {

**this**.lops = lops;

}

@Override

**public** String toString() {

**return** "Giaovien [maGiaovien=" + maGiaovien + ", ho=" + ho + ", ten=" + ten + "]";

}

}

**13. Dto layer:**

* **MonhocDTO**

**package** platform.web.springmvc.dto;

**public** **class** MonhocDTO {

**private** Integer maMonhoc;

**private** String tenMonhoc;

**// contructor & getter - setter**

}

* **LopDTO**

**package** platform.web.springmvc.dto;

**import** java.util.Date;

**import** com.fasterxml.jackson.annotation.JsonFormat;

**public** **class** LopDTO {

**private** Integer maLop;

**private** String tenLop;

@JsonFormat(shape = JsonFormat.Shape.***STRING***, pattern = "yyyy-MM-dd HH:mm a z")

**private** Date ngayBatdau;

@JsonFormat(shape = JsonFormat.Shape.***STRING***, pattern = "yyyy-MM-dd HH:mm a z")

**private** Date ngayKetthuc;

**private** **int** soGioHoc;

// contructor & getter - setter

}

* **@JsonFormat annotation:** là
* **SinhvienDTO**

**package** platform.web.springmvc.dto;

**public** **class** SinhvienDTO{

**private** **int** maSinhvien;

**private** String ho;

**private** String ten;

**// contructor & getter - setter**

**}**

* **GiaovienDTO**

**package** platform.web.springmvc.dto;

**public** **class** GiaovienDTO {

**private** Integer maGiaovien;

**private** String ho;

**private** String ten;

// contructor & getter - setter

}

**14. Kiểm tra kết quả:**

