#### 2数据库设计、开发环境配置、永久层及业务层设计

###### 2.1数据库设计

2.1.1概述

（1）数据库设计起源于数据流图，根据数据流图中的数据流向、变化及存储情况，构造概念模型及关系模型。

（2）本次实践使用MySQL数据库管理系统存储和管理系统数据。

2.1.2参考案例

1）SQL脚本（基于MySQL）

|  |
| --- |
| -- course管理平台  create database coursedb character set gbk;  use coursedb;  -- 设计客户端默认字符集  SET @saved\_cs\_client = @@character\_set\_client;  SET @@character\_set\_client = gbk;  CREATE TABLE IF NOT EXISTS `campus\_news` (  `id` BIGINT NOT NULL AUTO\_INCREMENT COMMENT "id",  `name` VARCHAR(100) NOT NULL COMMENT "用户",  `img\_url` VARCHAR(200) COMMENT "图片的url",  `forward\_url` VARCHAR(200) COMMENT "跳转的url",  PRIMARY KEY (`id`)  ) ENGINE=InnoDB COMMENT="校园新闻";  CREATE TABLE IF NOT EXISTS `user` (  `id` BIGINT NOT NULL AUTO\_INCREMENT COMMENT "id",  `name` VARCHAR(100) NOT NULL COMMENT "用户",  `role` TINYINT NOT NULL DEFAULT 0 COMMENT "用户的角色",  `department` VARCHAR(100) COMMENT "学院",  `major` VARCHAR(100) COMMENT "专业",  `number` VARCHAR(100) COMMENT "学号或教师号",  PRIMARY KEY (`id`)  ) ENGINE=InnoDB COMMENT="用户表";  CREATE TABLE IF NOT EXISTS `resource\_right\_to\_audit` (  `id` BIGINT NOT NULL AUTO\_INCREMENT COMMENT "id",  `user\_id` BIGINT NOT NULL COMMENT "可以审核资源的用户ID",  `resource\_id` BIGINT NOT NULL COMMENT "资源",  PRIMARY KEY (`id`)  ) ENGINE=InnoDB COMMENT="物资";  CREATE TABLE IF NOT EXISTS `resource` (  `id` BIGINT NOT NULL AUTO\_INCREMENT COMMENT "id",  `user\_id` BIGINT NOT NULL COMMENT "拥有这个资源的用户ID",  `name` VARCHAR(100) NOT NULL COMMENT "资源名",  `description` VARCHAR(300) COMMENT "资源描述",  `amount` BIGINT DEFAULT 0 NOT NULL COMMENT "资源的数量",  `effect\_start\_time` DATETIME NOT NULL COMMENT "有效期开始时间",  `effect\_end\_time` DATETIME NOT NULL COMMENT "有效期结束时间",  PRIMARY KEY (`id`)  ) ENGINE=InnoDB COMMENT="物资";  CREATE TABLE IF NOT EXISTS `resource\_change` (  `id` BIGINT NOT NULL AUTO\_INCREMENT COMMENT "id",  `user\_id` BIGINT NOT NULL COMMENT "操作这个资源的用户ID",  `resource\_id` BIGINT NOT NULL COMMENT "资源对应的ID",  `description` VARCHAR(300) COMMENT "资源去向或者增加描述",  `change\_amount` Integer NOT NULL COMMENT "资源的改变的数量",  `operate\_time` DATETIME NOT NULL COMMENT "有效期开始时间",  `status` TINYINT NOT NULL DEFAULT 0 COMMENT "审核的状态 0：未审核 1：批准 3 ：拒绝",  `status\_description` VARCHAR(300) COMMENT "审核描述",  PRIMARY KEY (`id`)  ) ENGINE=InnoDB COMMENT="物资";  CREATE TABLE IF NOT EXISTS `reimbursement` (  `id` BIGINT NOT NULL AUTO\_INCREMENT COMMENT "id",  `user\_id` BIGINT NOT NULL COMMENT "报销用户的ID",  `reimbursement\_type` TINYINT NOT NULL COMMENT "报销类型",  `event` VARCHAR(100) NOT NULL COMMENT "报销的具体事宜",  `pay\_way` TINYINT NOT NULL COMMENT "支付的方式",  `pay\_detail` VARCHAR(100) NOT NULL COMMENT "支付方式的信息",  `prove\_type` TINYINT NOT NULL COMMENT "证明材料类型",  `prove\_detail` TEXT NOT NULL COMMENT "证明材料的具体信息",  `event\_start\_time` DATETIME COMMENT "具体的开始时间点",  `event\_end\_time` DATETIME COMMENT "具体的结束时间点",  PRIMARY KEY (`id`)  ) ENGINE=InnoDB COMMENT="报销";  CREATE TABLE IF NOT EXISTS `classroom` (  `id` BIGINT NOT NULL AUTO\_INCREMENT COMMENT "id",  `name` VARCHAR(100) NOT NULL COMMENT "课室名",  `location\_first` TINYINT NOT NULL COMMENT "课室大位置",  `location\_second` Integer NOT NULL COMMENT "课室小位置",  `capacity` Integer NOT NULL COMMENT "容量",  PRIMARY KEY (`id`)  ) ENGINE=InnoDB COMMENT="课室";  CREATE TABLE IF NOT EXISTS `classroom\_present\_request` (  `id` BIGINT NOT NULL AUTO\_INCREMENT COMMENT "id",  `class\_id` BIGINT NOT NULL COMMENT "想要申请教室的ID",  `user\_id` BIGINT NOT NULL COMMENT "申请的用户ID",  `description` VARCHAR(300) COMMENT "申请的事宜描述",  `present\_day` DATETIME NOT NULL COMMENT "具体的开始时间点",  `start\_time` Integer NOT NULL COMMENT "具体的开始时间点",  `end\_time` Integer NOT NULL COMMENT "具体的结束时间点",  `prove\_type` TINYINT NOT NULL COMMENT "证明材料类型",  `prove\_detail` TEXT NOT NULL COMMENT "证明材料的具体信息",  PRIMARY KEY (`id`)  ) ENGINE=InnoDB COMMENT="课室申请"; |

**常见MySQL命令用法：**

-- 启动服务：运行MySQL安装目录中的mysqld.exe程序，启动数据库服务。**注意，只有启动数据库服务后，才能登录数据库服务。**

-- 登录：在命令行提示符下执行：mysql -u root –p

-- 显示所有数据库：show databases;

-- 设置当前数据库：use coursedb;

-- 显示所有表对象：show tables;

-- 显示表的结构信息：desc course;

-- 执行SQL脚本命令：source c:/coursedb.sql;

-- 显示环境变量值：show variables like "%character%";

-- 设置环境变量值：set character\_set\_client=gbk;

###### 2.2开发环境配置（Eclipse环境）

2.2.1 基础Jar包准备（按住Ctrl键，单击下列链接可进入下载页面）

1．Sping框架所需的Jar包

<http://repo.spring.io/simple/libs-release-local/org/springframework/spring/>

2．MyBatis框架所需的Jar包

<https://github.com/mybatis/mybatis-3/releases>

3．commons logging包

<http://commons.apache.org/proper/commons-logging/download_logging.cgi>

4．MyBatis与Spring整合的中间Jar包

<https://mvnrepository.com/artifact/org.mybatis/mybatis-spring>

5．MySQL数据库驱动所需Jar包

<https://dev.mysql.com/downloads/connector/j/5.1.html>

6．DBCP连接池Jar包

<http://commons.apache.org/proper/commons-dbcp/download_dbcp.cgi>

7．JSTL包

<http://archive.apache.org/dist/jakarta/taglibs/standard/binaries/>

2.2.2 SSM框架整合

（1）安装Tomcat和MySQL，并确保其可正常工作。

（2）在Eclipse中新建Web项目，加入Tomcat Web服务环境。

（3）将所需Jar包复制到/WEB-INF/lib文件夹中。

（4）在/WEB-INF文件夹中，新建web.xml文件，内容如下：

|  |
| --- |
| <?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>course</display-name>  <welcome-file-list>  <welcome-file>index.jsp</welcome-file>  </welcome-file-list>  <!-- 配置加载Spring文件的监听器-->  <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:config/applicationContext.xml</param-value>  </context-param>  <listener>  <listener-class>  org.springframework.web.context.ContextLoaderListener  </listener-class>  </listener>  <!-- 编码过滤器 -->  <filter>  <filter-name>encoding</filter-name>  <filter-class>  org.springframework.web.filter.CharacterEncodingFilter  </filter-class>  <init-param>  <param-name>encoding</param-name>  <param-value>UTF-8</param-value>  </init-param>  </filter>  <filter-mapping>  <filter-name>encoding</filter-name>  <url-pattern>\*</url-pattern>  </filter-mapping>  <!--Spring MVC前端器 -->  <servlet>  <servlet-name>course</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:config/springmvc-config.xml</param-value>  </init-param>  <load-on-startup>1</load-on-startup>  </servlet>  <servlet-mapping>  <servlet-name>course</servlet-name>  <url-pattern>/</url-pattern>  </servlet-mapping>  </web-app> |

（5）在src目录下，分别创建数据库常量配置文件db.properties、Spring配置文件applicationContext.xml、springmvc-config.xml和MyBatis配置文件mybatis-config.xml。三个文件的内容如下：

|  |
| --- |
| db.properties |
| jdbc.driver=com.mysql.jdbc.Driver  jdbc.url=jdbc:mysql://localhost:3306/resoucre?characterEncoding=GBK  jdbc.username=root  jdbc.password=qq123456  jdbc.maxTotal=30  jdbc.maxIdle=10  jdbc.initialSize=5 |
| applicationContext.xml |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xmlns:aop=*"http://www.springframework.org/schema/aop"*  xmlns:tx=*"http://www.springframework.org/schema/tx"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans-4.3.xsd*  *http://www.springframework.org/schema/tx*  *http://www.springframework.org/schema/tx/spring-tx-4.3.xsd*  *http://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context-4.3.xsd*  *http://www.springframework.org/schema/aop*  *http://www.springframework.org/schema/aop/spring-aop-4.3.xsd"*>  <!-- 读取db.properties -->  <context:property-placeholder location=*"classpath:config/db.properties"*/>  <!-- 配置数据源 -->  <bean id=*"dataSource"* class=*"org.apache.commons.dbcp2.BasicDataSource"*>  <!--数据库驱动 -->  <property name=*"driverClassName"* value=*"${jdbc.driver}"* />  <!--连接数据库的url -->  <property name=*"url"* value=*"${jdbc.url}"* />  <!--连接数据库的用户名 -->  <property name=*"username"* value=*"${jdbc.username}"* />  <!--连接数据库的密码 -->  <property name=*"password"* value=*"${jdbc.password}"* />  <!--最大连接数 -->  <property name=*"maxTotal"* value=*"${jdbc.maxTotal}"* />  <!--最大空闲连接 -->  <property name=*"maxIdle"* value=*"${jdbc.maxIdle}"* />  <!--初始化连接数 -->  <property name=*"initialSize"* value=*"${jdbc.initialSize}"* />  </bean>  <!-- 事务管理器，依赖于数据源 -->  <bean id=*"transactionManager"* class=  *"org.springframework.jdbc.datasource.DataSourceTransactionManager"*>  <property name=*"dataSource"* ref=*"dataSource"* />  </bean>  <!-- 开启事务注解 -->  <tx:annotation-driven transaction-manager=*"transactionManager"*/>  <!-- 配置MyBatis工厂SqlSessionFactory -->  <bean id=*"sqlSessionFactory"* class=*"org.mybatis.spring.SqlSessionFactoryBean"*>  <!--注入数据源 -->  <property name=*"dataSource"* ref=*"dataSource"* />  <!--指定核MyBatis心配置文件位置 -->  <property name=*"configLocation"* value=*"classpath:config/mybatis-config.xml"* />  </bean>  <!-- 配置mapper扫描器 -->  <bean class=*"org.mybatis.spring.mapper.MapperScannerConfigurer"*>  <property name=*"basePackage"* value=*"com.mapper"*/>  </bean>  <!-- 扫描Service -->  <context:component-scan base-package=*"com.service"* />  </beans> |
| springmvc-config.xml |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xmlns:mvc=*"http://www.springframework.org/schema/mvc"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xmlns:tx=*"http://www.springframework.org/schema/tx"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans-4.3.xsd*  *http://www.springframework.org/schema/mvc*  *http://www.springframework.org/schema/mvc/spring-mvc-4.3.xsd*  *http://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context-4.3.xsd"*>    <!-- 配置包扫描器，扫描@Controller注解的类 -->  <context:component-scan base-package=*"com.controller"* />  <!-- 加载注解驱动 -->  <mvc:annotation-driven />  <!--配置静态资源的访问映射，此配置中的文件，将不被前端控制器拦截 -->  <mvc:resources location=*"/css/"* mapping=*"/css/\*\*"*/>  <mvc:resources location=*"/fonts/"* mapping=*"/fonts/\*\*"*/>  <mvc:resources location=*"/images/"* mapping=*"/images/\*\*"*/>  <mvc:resources location=*"/js/"* mapping=*"/js/\*\*"*/>  <mvc:resources location=*"/uploadfiles/"* mapping=*"/uploadfiles/\*\*"*/>  <mvc:resources location=*"/videos/"* mapping=*"/videos/\*\*"*/>  <mvc:resources location=*"/files/"* mapping=*"/files/\*\*"*/>  <!-- 配置视图解析器 -->  <bean class=  *"org.springframework.web.servlet.view.InternalResourceViewResolver"*>  <property name=*"prefix"* value=*"/WEB-INF/jsp/"* />  <property name=*"suffix"* value=*".jsp"* />  </bean>  <bean id=*"multipartResolver"* class=*"org.springframework.web.multipart.commons.CommonsMultipartResolver"*>  <property name=*"defaultEncoding"* value=*"UTF-8"*/>  <property name=*"maxUploadSize"* value=*"2147483647"*></property>  </bean>    <!-- 配置拦截器 -->  <mvc:interceptors>  <mvc:interceptor>  <mvc:mapping path=*"/\*\*"* />  <mvc:exclude-mapping path=*"/css/\*\*"*/>  <mvc:exclude-mapping path=*"/fonts/\*\*"*/>  <mvc:exclude-mapping path=*"/images/\*\*"*/>  <mvc:exclude-mapping path=*"/js/\*\*"*/>  <mvc:exclude-mapping path=*"/files/\*\*"*/>  <bean class=*"com.interceptor.LoginInterceptor"* />  </mvc:interceptor>  </mvc:interceptors>  </beans> |
| mybatis-config.xml |
| <?xml version=*"1.0"* encoding=*"UTF-8"* ?>  <!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-config.dtd">  <configuration>  <settings>  <!-- 使用驼峰命名法转换字段。 -->  <setting name=*"mapUnderscoreToCamelCase"* value=*"true"*/>  </settings>  <!-- 别名定义 -->  <typeAliases>  <package name=*"com.po"* />  </typeAliases>  <plugins>  <plugin interceptor=*"com.github.pagehelper.PageHelper"*>  <property name=*"dialect"* value=*"mysql"* />  <property name=*"offsetAsPageNum"* value=*"false"* />  <property name=*"rowBoundsWithCount"* value=*"false"* />  <property name=*"pageSizeZero"* value=*"true"* />  <property name=*"reasonable"* value=*"false"* />  <property name=*"supportMethodsArguments"* value=*"false"* />  <property name=*"returnPageInfo"* value=*"none"* />  </plugin>  </plugins>  </configuration> |

###### 2.3持久层设计

持久层设计主要包括JavaBean设计和面向MyBatis的接口及映射文件两个部分。

常见的JavaBean包括：CampusNews，Classroom，ClassroomPresentRequest，Reimbursement，Resource，ResourceChange，ResourceRightToAudit，User等。

可先按JavaBean设计接口和映射文件，再根据业务流程要求，进一步优化和补充接口和映射文件中的内容。

###### 2.4业务层设计（以“课程单元”为例）

业务层主要提供基于数据库的业务流程操作，通常包括接口定义和实现两个部分。

如果控制器在处理请求时，无法简单地通过调用MyBatis接口响应请求，可考虑创建业务层类型，在 MyBatis接口的基础上，进一步丰富和扩展复杂的业务功能。注意，不能在控制器中直接编写数据处理（或业务处理）代码。

|  |
| --- |
| 业务层代码示例 |
| **public interface IService<T> {**  **int DEFAULT\_BATCH\_SIZE = 1000;**  **default boolean save(T entity) {**  **return SqlHelper.retBool(this.getBaseMapper().insert(entity));**  **}**  **@Transactional(**  **rollbackFor = {Exception.class}**  **)**  **default boolean saveBatch(Collection<T> entityList) {**  **return this.saveBatch(entityList, 1000);**  **}**  **boolean saveBatch(Collection<T> entityList, int batchSize);**  **@Transactional(**  **rollbackFor = {Exception.class}**  **)**  **default boolean saveOrUpdateBatch(Collection<T> entityList) {**  **return this.saveOrUpdateBatch(entityList, 1000);**  **}**  **boolean saveOrUpdateBatch(Collection<T> entityList, int batchSize);**  **default boolean removeById(Serializable id) {**  **return SqlHelper.retBool(this.getBaseMapper().deleteById(id));**  **}**  **default boolean removeByMap(Map<String, Object> columnMap) {**  **Assert.notEmpty(columnMap, "error: columnMap must not be empty", new Object[0]);**  **return SqlHelper.retBool(this.getBaseMapper().deleteByMap(columnMap));**  **}**  **default boolean remove(Wrapper<T> queryWrapper) {**  **return SqlHelper.retBool(this.getBaseMapper().delete(queryWrapper));**  **}**  **default boolean removeByIds(Collection<? extends Serializable> idList) {**  **return CollectionUtils.isEmpty(idList) ? false : SqlHelper.retBool(this.getBaseMapper().deleteBatchIds(idList));**  **}**  **default boolean updateById(T entity) {**  **return SqlHelper.retBool(this.getBaseMapper().updateById(entity));**  **}**  **default boolean update(Wrapper<T> updateWrapper) {**  **return this.update((Object)null, updateWrapper);**  **}**  **default boolean update(T entity, Wrapper<T> updateWrapper) {**  **return SqlHelper.retBool(this.getBaseMapper().update(entity, updateWrapper));**  **}**  **@Transactional(**  **rollbackFor = {Exception.class}**  **)**  **default boolean updateBatchById(Collection<T> entityList) {**  **return this.updateBatchById(entityList, 1000);**  **}**  **boolean updateBatchById(Collection<T> entityList, int batchSize);**  **boolean saveOrUpdate(T entity);**  **default T getById(Serializable id) {**  **return this.getBaseMapper().selectById(id);**  **}**  **default List<T> listByIds(Collection<? extends Serializable> idList) {**  **return this.getBaseMapper().selectBatchIds(idList);**  **}**  **default List<T> listByMap(Map<String, Object> columnMap) {**  **return this.getBaseMapper().selectByMap(columnMap);**  **}**  **default T getOne(Wrapper<T> queryWrapper) {**  **return this.getOne(queryWrapper, true);**  **}**  **T getOne(Wrapper<T> queryWrapper, boolean throwEx);**  **Map<String, Object> getMap(Wrapper<T> queryWrapper);**  **<V> V getObj(Wrapper<T> queryWrapper, Function<? super Object, V> mapper);**  **default int count() {**  **return this.count(Wrappers.emptyWrapper());**  **}**  **default int count(Wrapper<T> queryWrapper) {**  **return SqlHelper.retCount(this.getBaseMapper().selectCount(queryWrapper));**  **}**  **default List<T> list(Wrapper<T> queryWrapper) {**  **return this.getBaseMapper().selectList(queryWrapper);**  **}**  **default List<T> list() {**  **return this.list(Wrappers.emptyWrapper());**  **}**  **default <E extends IPage<T>> E page(E page, Wrapper<T> queryWrapper) {**  **return this.getBaseMapper().selectPage(page, queryWrapper);**  **}**  **default <E extends IPage<T>> E page(E page) {**  **return this.page(page, Wrappers.emptyWrapper());**  **}**  **default List<Map<String, Object>> listMaps(Wrapper<T> queryWrapper) {**  **return this.getBaseMapper().selectMaps(queryWrapper);**  **}**  **default List<Map<String, Object>> listMaps() {**  **return this.listMaps(Wrappers.emptyWrapper());**  **}**  **default List<Object> listObjs() {**  **return this.listObjs(Function.identity());**  **}**  **default <V> List<V> listObjs(Function<? super Object, V> mapper) {**  **return this.listObjs(Wrappers.emptyWrapper(), mapper);**  **}**  **default List<Object> listObjs(Wrapper<T> queryWrapper) {**  **return this.listObjs(queryWrapper, Function.identity());**  **}**  **default <V> List<V> listObjs(Wrapper<T> queryWrapper, Function<? super Object, V> mapper) {**  **return (List)this.getBaseMapper().selectObjs(queryWrapper).stream().filter(Objects::nonNull).map(mapper).collect(Collectors.toList());**  **}**  **default <E extends IPage<Map<String, Object>>> E pageMaps(E page, Wrapper<T> queryWrapper) {**  **return this.getBaseMapper().selectMapsPage(page, queryWrapper);**  **}**  **default <E extends IPage<Map<String, Object>>> E pageMaps(E page) {**  **return this.pageMaps(page, Wrappers.emptyWrapper());**  **}**  **BaseMapper<T> getBaseMapper();**  **Class<T> getEntityClass();**  **default QueryChainWrapper<T> query() {**  **return ChainWrappers.queryChain(this.getBaseMapper());**  **}**  **default LambdaQueryChainWrapper<T> lambdaQuery() {**  **return ChainWrappers.lambdaQueryChain(this.getBaseMapper());**  **}**  **default KtQueryChainWrapper<T> ktQuery() {**  **return ChainWrappers.ktQueryChain(this.getBaseMapper(), this.getEntityClass());**  **}**  **default KtUpdateChainWrapper<T> ktUpdate() {**  **return ChainWrappers.ktUpdateChain(this.getBaseMapper(), this.getEntityClass());**  **}**  **default UpdateChainWrapper<T> update() {**  **return ChainWrappers.updateChain(this.getBaseMapper());**  **}**  **default LambdaUpdateChainWrapper<T> lambdaUpdate() {**  **return ChainWrappers.lambdaUpdateChain(this.getBaseMapper());**  **}**  **default boolean saveOrUpdate(T entity, Wrapper<T> updateWrapper) {**  **return this.update(entity, updateWrapper) || this.saveOrUpdate(entity);**  **}**  **}**  **// 接口实现代码**  package com.florence.resources.service.impl;  import com.florence.resources.po.CampusNews;  import com.florence.resources.mapper.CampusNewsMapper;  import com.florence.resources.service.ICampusNewsService;  import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;  import org.springframework.stereotype.Service;  /\*\*  \* <p>  \* 校园新闻 服务实现类  \* </p>  \*  \* @author Florence  \* @since 2021-10-30  \*/  @Service  public class CampusNewsServiceImpl extends ServiceImpl<CampusNewsMapper, CampusNews> implements ICampusNewsService {  }  package com.florence.resources.service.impl;  import com.florence.resources.po.ClassroomPresentRequest;  import com.florence.resources.mapper.ClassroomPresentRequestMapper;  import com.florence.resources.service.IClassroomPresentRequestService;  import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;  import org.springframework.stereotype.Service;  /\*\*  \* <p>  \* 课室申请 服务实现类  \* </p>  \*  \* @author Florence  \* @since 2021-10-30  \*/  @Service  public class ClassroomPresentRequestServiceImpl extends ServiceImpl<ClassroomPresentRequestMapper, ClassroomPresentRequest> implements IClassroomPresentRequestService {  }  package com.florence.resources.service.impl;  import com.florence.resources.po.Classroom;  import com.florence.resources.mapper.ClassroomMapper;  import com.florence.resources.service.IClassroomService;  import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;  import org.springframework.stereotype.Service;  /\*\*  \* <p>  \* 课室 服务实现类  \* </p>  \*  \* @author Florence  \* @since 2021-10-30  \*/  @Service  public class ClassroomServiceImpl extends ServiceImpl<ClassroomMapper, Classroom> implements IClassroomService {  }  package com.florence.resources.service.impl;  import com.florence.resources.po.Reimbursement;  import com.florence.resources.mapper.ReimbursementMapper;  import com.florence.resources.service.IReimbursementService;  import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;  import org.springframework.stereotype.Service;  /\*\*  \* <p>  \* 报销 服务实现类  \* </p>  \*  \* @author Florence  \* @since 2021-10-30  \*/  @Service  public class ReimbursementServiceImpl extends ServiceImpl<ReimbursementMapper, Reimbursement> implements IReimbursementService {  }  package com.florence.resources.service.impl;  import com.florence.resources.po.ResourceChange;  import com.florence.resources.mapper.ResourceChangeMapper;  import com.florence.resources.service.IResourceChangeService;  import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;  import org.springframework.stereotype.Service;  /\*\*  \* <p>  \* 物资装填 服务实现类  \* </p>  \*  \* @author Florence  \* @since 2021-10-30  \*/  @Service  public class ResourceChangeServiceImpl extends ServiceImpl<ResourceChangeMapper, ResourceChange> implements IResourceChangeService {  }  package com.florence.resources.service.impl;  import com.florence.resources.po.ResourceRightToAudit;  import com.florence.resources.mapper.ResourceRightToAuditMapper;  import com.florence.resources.service.IResourceRightToAuditService;  import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;  import org.springframework.stereotype.Service;  /\*\*  \* <p>  \* 物资 服务实现类  \* </p>  \*  \* @author Florence  \* @since 2021-10-30  \*/  @Service  public class ResourceRightToAuditServiceImpl extends ServiceImpl<ResourceRightToAuditMapper, ResourceRightToAudit> implements IResourceRightToAuditService {  }  package com.florence.resources.service.impl;  import com.florence.resources.po.Resource;  import com.florence.resources.mapper.ResourceMapper;  import com.florence.resources.service.IResourceService;  import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;  import org.springframework.stereotype.Service;  /\*\*  \* <p>  \* 物资 服务实现类  \* </p>  \*  \* @author Florence  \* @since 2021-10-30  \*/  @Service  public class ResourceServiceImpl extends ServiceImpl<ResourceMapper, Resource> implements IResourceService {  }  package com.florence.resources.service.impl;  import com.florence.resources.po.User;  import com.florence.resources.mapper.UserMapper;  import com.florence.resources.service.IUserService;  import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;  import org.springframework.stereotype.Service;  /\*\*  \* <p>  \* 用户表 服务实现类  \* </p>  \*  \* @author Florence  \* @since 2021-10-30  \*/  @Service  public class UserServiceImpl extends ServiceImpl<UserMapper, User> implements IUserService {  } |