#### 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（线上课程学习平台）

1）概念模型（E-R图）



图1 线上课程学习平台数据库E-R图

2）关系模型

系统选项：sys\_keycode(key\_name,key\_value)

教师：teacher(tid,tname,email,mobile,password,img,register\_date,update\_date, school,department,major)

开课学期：term(term\_id,term\_name)

课程信息：course\_info(cid,cname,description,preknowledge,outline,reference, brief,cover,term\_id,start\_date,end\_date,tid)

课程学生信息：student\_register(srid,cid,sno,sname,is\_registered)

学生：student(sid,email,password,img,register\_date)

学生申请的课程：course\_student(srid,sid,cid,apply\_date)

角色：role(role\_id,role\_name,detail)

公告：bulletin(bid,content,pub\_date,tid,cid)

课程附件类型：course\_file\_type(type\_id,type\_name)

课程附件：course\_file(fid,cid,file\_name,file\_path,type\_id,upload\_date,brief)

视频文件：course\_video(vid,cid,title,file\_name,upload\_date,brief)

单元类型：unit\_type(type\_id,type\_name)

教学单元：course\_unit(uid,uname,brief,type\_id,cid,visible,end\_time)

教学单元视频：unit\_video(uid,vid)

单元学习进度：unit\_student(usid,vid,uid,sid,study\_time,progress)

单元成绩：unit\_grade(uid,sid,grade)

3）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;  -- 系统部分选项值，如课程代码的后9位，用于在生成课程代码参考  create table sys\_keycode(  key\_name varchar(30) primary key,  key\_value varchar(30) not null  )ENGINE=InnoDB DEFAULT CHARSET=gbk;  set names gbk;  insert into sys\_keycode values('user\_image','images/user.png'); -- 用户默认头像  insert into sys\_keycode values('course\_image','images/course\_default.jpg');  -- 教师  create table teacher(  tid int primary key auto\_increment,  tname varchar(30) not null,  email varchar(30) not null, -- 登录时，用邮箱  mobile varchar(20),  password varchar(30) not null,  img varchar(256), -- 图像  registerDate date,  updateDate date,  school varchar(50),  department varchar(50),  major varchar(50),  unique(email)  )ENGINE=InnoDB DEFAULT CHARSET=gbk;  set names gbk;  insert into teacher(password,tname,tsex,email,mobile,depart,img) values('900150983CD24FB0D6963F7D28E17F72','张三丰','男','abc@qq.com','13535215175','广东金融学院','images/abc.jpg');  -- 开课学期  create table term(  term\_id int primary key auto\_increment,  term\_name char(30) not null  )ENGINE=InnoDB DEFAULT CHARSET=gbk;  set names gbk;  insert into term(term\_name) values('2020上学期');  insert into term(term\_name) values('2020下学期');  insert into term(term\_name) values('2021上学期');  insert into term(term\_name) values('2020下学期');  -- 课程信息  create table course\_info(  cid char(14) primary key, -- 前3位为类别，中8位为申请日期，后3位为编号  cname varchar(50) not null,  description text,  preknowledge text,  outline text,  reference text,  brief text, -- 简介  cover varchar(256), -- 封面图片  term\_id int, -- 开课学期  start\_date datetime, -- 起始日期  end\_date datetime, -- 终止日期  tid int, -- 创建者  constraint fk\_course\_info\_term foreign key(term\_id) references term(term\_id),  constraint fk\_course\_info\_teacher foreign key(tid) references teacher(tid)  )ENGINE=InnoDB DEFAULT CHARSET=gbk;  -- 在课程中导入的学生基础信息，便于成绩管理  create table student\_register(  srid int primary key auto\_increment,  cid char(14) not null,  sno varchar(30) not null,  sname varchar(30) not null,  is\_registered int,  constraint fk\_sr\_course foreign key(cid) references course\_info(cid)  )ENGINE=InnoDB DEFAULT CHARSET=gbk;  -- 学生  create table student(  sid int primary key auto\_increment,  email varchar(30), -- 登录用  password varchar(30),  img varchar(256), -- 图像  register\_date date,  unique(email)  )ENGINE=InnoDB DEFAULT CHARSET=gbk;  -- 学生申请的课程  create table course\_student(  srid int not null,  sid int not null,  cid char(14) not null,  apply\_date date,  constraint pk\_cs primary key(sid,cid),  constraint fk\_cs\_course foreign key(cid) references course\_info(cid),  constraint fk\_cs\_register foreign key(srid) references student\_register(srid),  constraint fk\_cs\_student foreign key(sid) references student(sid)  )ENGINE=InnoDB DEFAULT CHARSET=gbk;  create table role(  role\_id int primary key auto\_increment,  role\_name varchar(30) not null,  detail varchar(50)  );ENGINE=InnoDB DEFAULT CHARSET=gbk;  set names gbk;  insert into role(role\_name) value('学生');  insert into role(role\_name) value('教师');  insert into role(role\_name) value('系统管理员');  -- 公告  create table bulletin(  bid int primary key auto\_increment,  content text not null,  pub\_date datetime not null,  tid int not null,  cid char(14) not null,  constraint fk\_bulletin\_teacher foreign key(tid) references teacher(tid),  constraint fk\_bulletin\_course\_info foreign key(cid) references course\_info(cid)  )ENGINE=InnoDB DEFAULT CHARSET=gbk;  -- 课程文件类型，包括课件，资料，视频等  create table course\_file\_type(  type\_id int primary key auto\_increment,  type\_name varchar(30)  )ENGINE=InnoDB DEFAULT CHARSET=gbk;  set names gbk;  insert into course\_file\_type(type\_name) values('课件');  insert into course\_file\_type(type\_name) values('教学基础文件');  insert into course\_file\_type(type\_name) values('其他资料');  -- 课程附件  create table course\_file(  fid int primary key auto\_increment,  cid char(14) not null,  file\_name varchar(100) not null,  file\_path varchar(255) not null,  type\_id int not null,  upload\_date date,  brief text,  constraint fk\_file\_type foreign key(type\_id) references course\_file\_type(type\_id)  )ENGINE=InnoDB DEFAULT CHARSET=gbk;  -- 视频文件  create table course\_video(  vid int primary key auto\_increment,  cid char(14) not null,  title varchar(100) not null,  file\_name varchar(256) not null,  upload\_date date,  brief text  )ENGINE=InnoDB DEFAULT CHARSET=gbk;  -- 单元类型  create table unit\_type(  type\_id int primary key auto\_increment,  type\_name varchar(30) not null  )ENGINE=InnoDB DEFAULT CHARSET=gbk;  set names gbk;  insert into unit\_type(type\_name) values('练习');  insert into unit\_type(type\_name) values('作业');  insert into unit\_type(type\_name) values('考试');  -- 课程教学单元，每门课相同  create table course\_unit(  uid int primary key auto\_increment,  uname varchar(30) not null,  brief text,  type\_id int not null,  cid char(14) not null,  visible int not null,  end\_time datetime, -- 对于练习类型的单元，不限制时间，其它类型的单元，到期后自动隐藏  constraint fk\_course\_unit\_course\_info foreign key(cid) references course\_info(cid),  constraint fk\_course\_unit\_type foreign key(type\_id) references unit\_type(type\_id)  )ENGINE=InnoDB DEFAULT CHARSET=gbk;  -- 单元内容  create table unit\_video(  uid int not null,  vid int not null,  constraint fk\_unit\_video\_course\_video foreign key(vid) references course\_video(vid),  constraint fk\_unit\_video\_course\_unit foreign key(uid) references course\_unit(uid),  constraint pk\_unit\_video primary key(uid,vid)  )ENGINE=InnoDB DEFAULT CHARSET=gbk;  -- 学生学习状态  create table unit\_student(  usid int primary key auto\_increment,  vid int not null,  uid int not null,  sid int not null,  study\_time datetime not null,  progress int not null default 0,  constraint fk\_unit\_student\_course\_video foreign key(vid) references course\_video(vid),  constraint fk\_unit\_student\_course\_unit foreign key(uid) references course\_unit(uid),  constraint fk\_unit\_student\_student foreign key(sid) references student(sid)  )ENGINE=InnoDB DEFAULT CHARSET=gbk;  -- 单元成绩  create table unit\_grade(  uid int not null,  sid int not null,  grade int default 0,  constraint fk\_unit\_grade\_course\_unit foreign key(uid) references course\_unit(uid),  constraint fk\_unit\_grade\_student foreign key(sid) references student(sid),  constraint pk\_unit\_grade primary key(uid,sid)  )ENGINE=InnoDB DEFAULT CHARSET=gbk;  -- 试题部分（略）  -- 恢复客户端默认字符集  SET @@character\_set\_client = @saved\_cs\_client; |

**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/coursedb?characterEncoding=GBK  jdbc.username=root  jdbc.password=root  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：Bulletin，CourseFile，CourseFileType，CourseInfo，CourseUnit，CourseVideo，Term，UnitGrade，UnitStudent，UnitType，UnitVideo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 |
| **//业务接口代码**  package com.service.course;  import java.util.List;  import com.po.course.CourseUnit;  public interface CourseUnitService {  CourseUnit findCourseUnitByUid(int uid);  List<CourseUnit> findCourseUnitByCid(String cid);  List<CourseUnit> findCourseUnitBySidCid(int sid, String cid);  List<CourseUnit> getValidateCourseUnit(List<CourseUnit> courseUnits); // 可见且未过期  List<CourseUnit> getVisibleCourseUnit(List<CourseUnit> courseUnits); // 可见    int insertCourseUnit(CourseUnit courseUnit);  int insertUnitVideos(int[] vid, int uid);    int updateCourseUnit(CourseUnit courseUnit);    int deleteCourseUnitByUid(int uid);  int deleteCourseUnitVideoByUid(int uid);  int deleteCourseUnitVideoByVidUid(int[] vids, int uid);  } |
| **// 接口实现代码**  package com.service.course.impl;  import java.text.SimpleDateFormat;  import java.util.Date;  import java.util.List;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import org.springframework.transaction.annotation.Transactional;  import com.mapper.course.CourseUnitMapper;  import com.mapper.course.UnitStudentMapper;  import com.po.course.CourseUnit;  import com.po.course.InputParameter;  import com.po.course.UnitVideo;  import com.service.course.CourseUnitService;  @Service  @Transactional  public class CourseUnitServiceImpl implements CourseUnitService{  @Autowired  CourseUnitMapper courseUnitMapper;  @Autowired  UnitStudentMapper unitStudentMapper;    @Override  public CourseUnit findCourseUnitByUid(int uid) {  // TODO Auto-generated method stub  return courseUnitMapper.findCourseUnitByUid(uid);  }  @Override  public List<CourseUnit> findCourseUnitByCid(String cid) {  // TODO Auto-generated method stub  return courseUnitMapper.findCourseUnitByCid(cid);  }  @Override  public int insertCourseUnit(CourseUnit courseUnit) {  // TODO Auto-generated method stub  return courseUnitMapper.insertCourseUnit(courseUnit);  }  @Override  public int updateCourseUnit(CourseUnit courseUnit) {  // TODO Auto-generated method stub  return courseUnitMapper.updateCourseUnit(courseUnit);  }  @Override  public int deleteCourseUnitByUid(int uid) {  // TODO Auto-generated method stub  return courseUnitMapper.deleteCourseUnitByUid(uid);  }  @Override  public int deleteCourseUnitVideoByUid(int uid) {  // TODO Auto-generated method stub  int count = courseUnitMapper.deleteCourseUnitVideoByUid(uid);  if (count == 0)  return 0;  count = courseUnitMapper.deleteCourseUnitByUid(uid);  return count;  }  @Override  public int insertUnitVideos(int[] vid, int uid) {  // TODO Auto-generated method stub  UnitVideo unitVideo = new UnitVideo();  unitVideo.setUid(uid);  int count = 0;  for (int i = 0; i < vid.length; i ++) {  unitVideo.setVid(vid[i]);  count += courseUnitMapper.insertUnitVideo(unitVideo);  }  return count;  }  @Override  public List<CourseUnit> findCourseUnitBySidCid(int sid, String cid) {  // TODO Auto-generated method stub  InputParameter inputParameter = new InputParameter();  inputParameter.setCid(cid);  inputParameter.setSid(sid);  List<CourseUnit> completedQuests = courseUnitMapper.findCourseUnitByCidSid(inputParameter); // 学生已完成的任务（视频）数量  List<CourseUnit> courseUnits = courseUnitMapper.findCourseUnitByCid(cid); // 含所有任务（视频）数量    // 合并  for (int i = 0; i < completedQuests.size(); i ++) {  for (int j = 0; j < courseUnits.size(); j ++) {  if (completedQuests.get(i).getUid() == courseUnits.get(j).getUid()) {  courseUnits.get(j).setCompletedCount(completedQuests.get(i).getCompletedCount());  if (courseUnits.get(j).getQuestCount() > 0)  courseUnits.get(j).setPercent((int)(1.0 \* courseUnits.get(j).getCompletedCount() / courseUnits.get(j).getQuestCount() \* 100));  }  }  }  return courseUnits;  }  @Override  public List<CourseUnit> getValidateCourseUnit(List<CourseUnit> courseUnits) {  // TODO Auto-generated method stub  SimpleDateFormat sdf=new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");  String currTime = sdf.format(new Date());  int size = courseUnits.size();  for (int i = size - 1; i >= 0; i --) {  if (courseUnits.get(i).getVisible() == 0 || currTime.compareTo(courseUnits.get(i).getEndTime()) >= 0 )  courseUnits.remove(i);  }  return courseUnits;  }  @Override  public List<CourseUnit> getVisibleCourseUnit(List<CourseUnit> courseUnits) {  // TODO Auto-generated method stub  int size = courseUnits.size();  for (int i = size - 1; i >= 0; i --) {  if (courseUnits.get(i).getVisible() == 0)  courseUnits.remove(i);  }  return courseUnits;  }  @Override  public int deleteCourseUnitVideoByVidUid(int[] vids, int uid) {  // TODO Auto-generated method stub  int count = 0;  InputParameter inputParameter = new InputParameter();  inputParameter.setUid(uid);  for (int i = 0; i < vids.length; i ++) {  inputParameter.setVid(vids[i]);  count += unitStudentMapper.deleteUnitStudentByVidUid(inputParameter);  }  count = 0;  for (int i = 0; i < vids.length; i ++) {  inputParameter.setVid(vids[i]);  count += courseUnitMapper.deleteCourseUnitVideoByVidUid(inputParameter);  }  return count;  }  } |