

任务一：拉勾教育后台管理系统（SSM）

1. 项目架构

1.1 项目介绍

拉勾教育后台管理系统,是提供给拉勾教育的相关业务人员使用的一个后台管理系统, 业务人员可以在这个后台管理系统中,对课程信息、广告信息、用户信息、 权限信息等数据进行维护。

在 web阶段,我们已经完成了拉勾教育后台管理系统中课程模块, 接下来将对拉勾教育后台管理系统进行升级改造, 基于SSM框架来完成课程信息模块, 广告信息模块, 用户信息模块, 权限信息模块

1.2 页面原型展示

访问 <http://eduboss.lagou.com>

用户名：18201288771密码：111111



1.3 技术选型

1.3.1 前端技术选型

前端技术	说明
Vue.js	是一套用于构建用户界面的渐进式JavaScript框架
Element UI 库	element-ui 是饿了么前端出品的基于 Vue.js的 后台组件库, 方便程序员进行页面快速布局和构建
node.js	简单的说 Node.js 就是运行在服务端的 JavaScript 运行环境 .
axios	对ajax的封装, 简单来说就是ajax技术实现了局部数据的刷新, axios实现了对 ajax的封装,

1.3.2 后端技术选型

后端技术	说明
Web层	借助springmvc接收请求，进行视图跳转
Service层	借助spring进行IOC、AOP、及事务管理
dao层	借助mybatis进行数据库交互

1.4 项目开发环境

- 开发工具
 - 后端: IDEA 2019
 - 前端: VS code
 - 数据库客户端工具: SQLYog
- 开发环境
 - JDK 11
 - Maven 3.6.3
 - MySQL 5.7

2. Maven进阶使用（Maven聚合工程）

2.1 maven基础知识回顾

2.1.1 maven介绍

maven 是一个项目管理工具，主要作用是在项目开发阶段对Java项目进行依赖管理和项目构建。

依赖管理：就是对jar包的管理。通过导入maven坐标，就相当于将仓库中的jar包导入了当前项目中。

项目构建：通过maven的一个命令就可以完成项目从清理、编译、测试、报告、打包，部署整个过程。



2.1.2 maven的仓库类型

1.本地仓库

2.远程仓库

①maven中央仓库（地址：<http://repo2.maven.org/maven2/>）

②maven私服（公司局域网内的仓库，需要自己搭建）

③其他公共远程仓库（例如apache提供的远程仓库，地址：<http://repo.maven.apache.org/maven2/>）

本地仓库---》maven私服---》maven中央仓库

2.1.3 maven常用命令

clean: 清理

compile: 编译

test: 测试

package: 打包

install: 安装

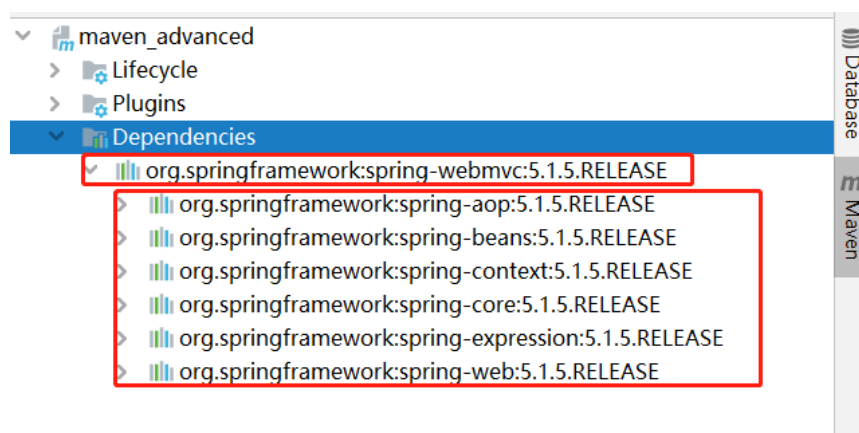
2.1.4 maven坐标书写规范

```
<dependency>
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
<version>5.1.32</version>
</dependency>
```

2.2 maven的依赖传递

2.2.1 什么是依赖传递

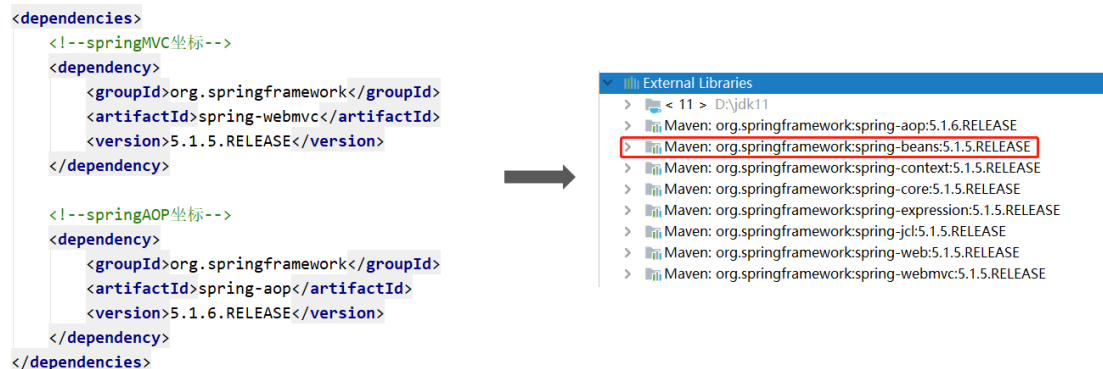
在maven中，依赖是可以传递的，假设存在三个项目，分别是项目A，项目B以及项目C。假设C依赖B，B依赖A，那么我们可以根据maven项目依赖的特征不难推出项目C也依赖A。



通过上面的图可以看到，我们的web项目直接依赖了spring-webmvc，而spring-webmvc依赖了spring-aop、spring-beans等。最终的结果就是在我们的web项目中间接依赖了spring-aop、spring-beans等。

依赖冲突

由于依赖传递现象的存在，spring-webmvc 依赖 spring-beans-5.1.5，spring-aop 依赖 spring-beans-5.1.6，但是发现 spring-beans-5.1.5 加入到了工程中，而我们希望 spring-beans-5.1.6 加入工程。这就造成了依赖冲突。



2.2.2 如何解决依赖冲突

1.使用maven提供的依赖调解原则

第一声明者优先原则

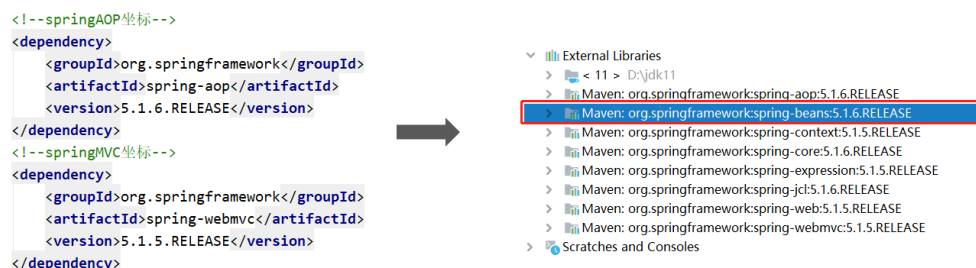
路径近者优先原则

2.排除依赖

3.锁定版本

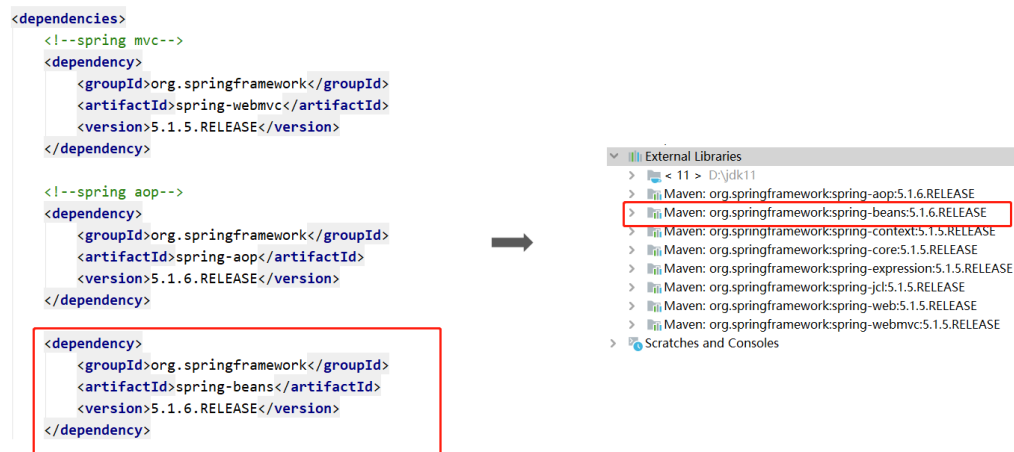
2.2.3 依赖调节原则——第一声明者优先原则

在 pom 文件中定义依赖，以先声明的依赖为准。其实就是根据坐标导入的顺序来确定最终使用哪个传递过来的依赖。



结论：通过上图可以看到，spring-aop和spring-webmvc都传递过来了spring-beans，但是因为spring-aop在前面，所以最终使用的spring-beans是由spring-aop传递过来的，而spring-webmvc传递过来的spring-beans则被忽略了。

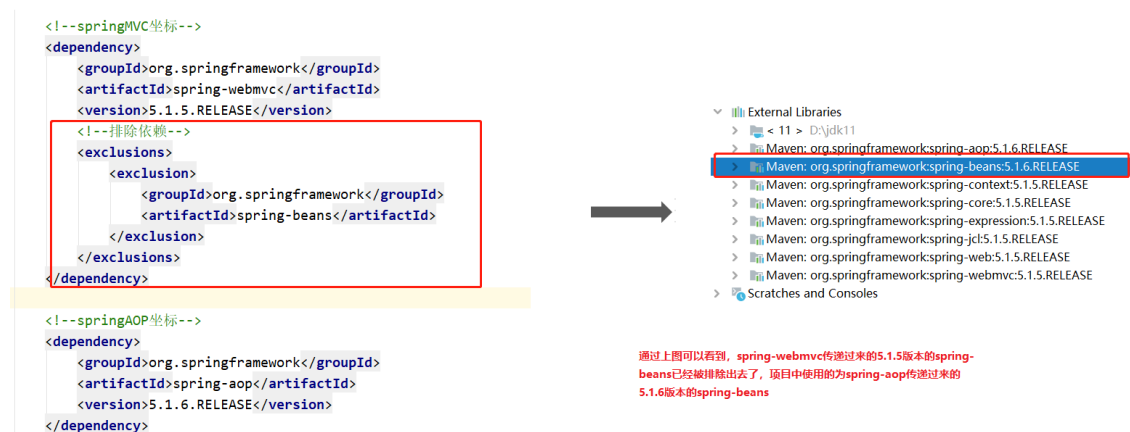
2.2.4 依赖调节原则——路径近者优先原则



总结：直接依赖大于依赖传递

2.2.5 排除依赖

可以使用exclusions标签将传递过来的依赖排除出去。



2.2.6 版本锁定

采用直接锁定版本的方法确定依赖jar包的版本，版本锁定后则不考虑依赖的声明顺序或依赖的路径，以锁定的版本为准添加到工程中，此方法在企业开发中经常使用。

版本锁定的使用方式：

第一步：在dependencyManagement标签中锁定依赖的版本

第二步：在dependencies标签中声明需要导入的maven坐标

①在dependencyManagement标签中锁定依赖的版本

```
<!-- 依赖jar包版本锁定 -->
```

```
<dependencyManagement>
```

```
<dependencies>
```

```
<dependency>
```

```
<groupId>org.springframework</groupId>
```

```
<artifactId>spring-beans</artifactId>
```

```
<version>5.1.6.RELEASE</version>
```

```
</dependency>
```

```
<dependency>
```

```
<groupId>org.springframework</groupId>
```

```
<artifactId>spring-context</artifactId>
```

```
<version>5.1.6.RELEASE</version>
```

```
</dependency>
```

```
</dependencies>
```

```
</dependencyManagement>
```

```
</dependencies>
```

注意：pom文件中使用dependencyManagement标签进行依赖jar的版本锁定，并不会真正将jar包导入到项目中，只是对这些jar的版本进行锁定。项目中使用那些jar包，还需要在dependencies标签中进行声明。

②在dependencies标签中声明需要导入的maven坐标

```
<dependencies>
```

```
<dependency>
```

```
<groupId>org.springframework</groupId>
```

```
<artifactId>spring-webmvc</artifactId>
```

```
<version>5.1.5.RELEASE</version>
```

```
</dependency>
```

```
<dependency>
```

```
<groupId>org.springframework</groupId>
```

```
<artifactId>spring-beans</artifactId>
```

```
</dependency>
```

```
<dependency>
```

```
<groupId>org.springframework</groupId>
```

```
<artifactId>spring-context</artifactId>
```

```
</dependency>
```

```
</dependencies>
```

注意：由于前面已经在dependencyManagement标签中锁定了依赖jar包的版本，后面需要导入依赖时只需要指定groupId和artifactId，无需再指定version。

2.2.7 properties标签的使用

```
<properties>
```

```
<spring.version>5.1.5.RELEASE</spring.version>
```

```
<springmvc.version>5.1.5.RELEASE</springmvc.version>
```

```
<mybatis.version>3.5.1</mybatis.version>
```

```
</properties>
```

```
<!-- 锁定jar版本 -->
```

```
<dependencyManagement>
```

```
<dependencies>
```

```
<!-- Mybatis -->
```

```
<dependency>
```

```
<groupId>org.mybatis</groupId>
```

```
<artifactId>mybatis</artifactId>
```

```
<version>${mybatis.version}</version>
```

```
</dependency>
```

```
<!-- springMVC -->
```

```
<dependency>
```

```
<groupId>org.springframework</groupId>
```

```
<artifactId>spring-webmvc</artifactId>
```

```
<version>${springmvc.version}</version>
```

```
</dependency>
```

```
<!-- spring -->
```

```
<dependency>
```

```
<groupId>org.springframework</groupId>
```

```
<artifactId>spring-context</artifactId>
```

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

```
</dependency>
```

```
<dependency>
```

```
<groupId>org.springframework</groupId>
```

```

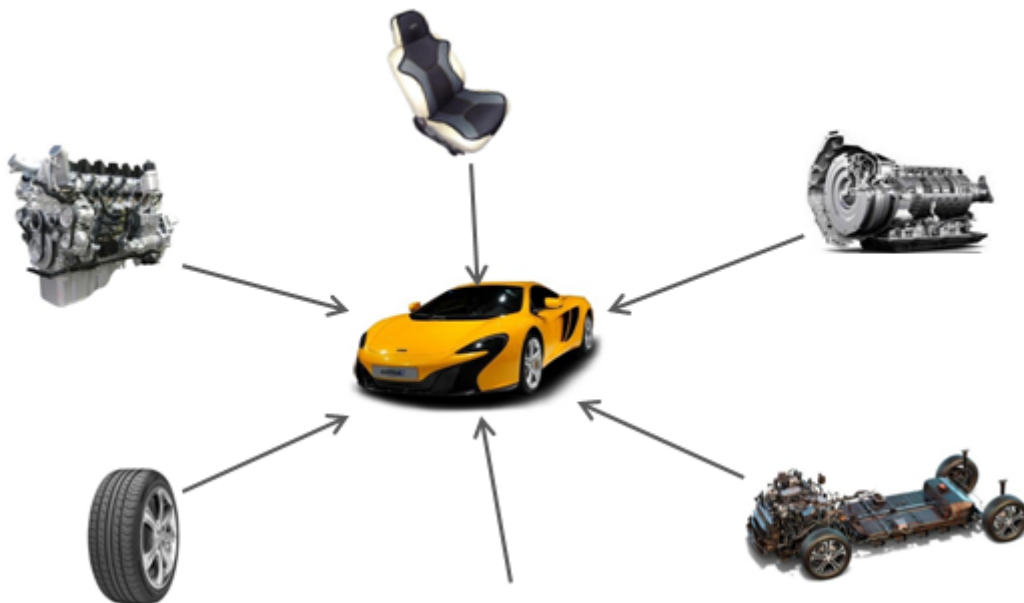
        <artifactId>spring-core</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-aop</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-web</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-expression</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-beans</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-aspects</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context-support</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-test</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-jdbc</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-tx</artifactId>
        <version>${spring.version}</version>
    </dependency>
</dependencies>
</dependencyManagement>

```

2.3 maven聚合工程（分模块）

概念:

在现实生活中，汽车厂家进行汽车生产时，由于整个生产过程非常复杂和繁琐，工作量非常大，所以厂家都会将整个汽车的部件分开生产，最终再将生产好的部件进行组装，形成一台完整的汽车。



2.3.1 分模块构建maven工程分析

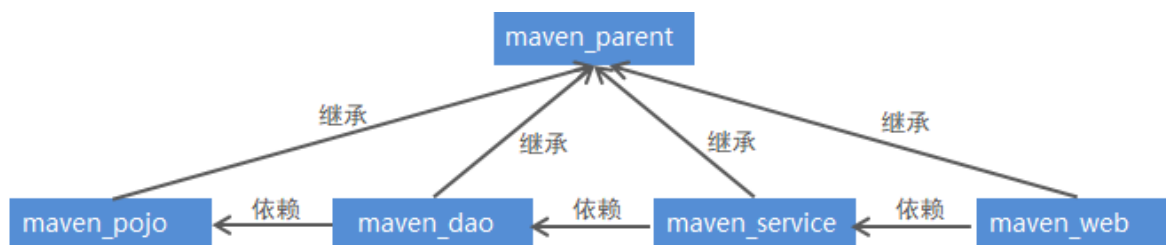
在企业项目开发中，由于项目规模大，业务复杂，参与的人员比较多，一般会通过合理的模块拆分将一个大型的项目拆分为N多个小模块，分别进行开发。而且拆分出的模块可以非常容易被其他模块复用

常见的拆分方式有两种：

第一种：按照**业务模块**进行拆分，每个模块拆分成一个maven工程，例如将一个项目分为用户模块，订单模块，购物车模块等，每个模块对应就是一个maven工程

第二种：按照**层**进行拆分，例如持久层、业务层、表现层等，每个层对应就是一个maven工程

不管上面那种拆分方式，通常都会提供一个父工程，将一些公共的代码和配置提取到父工程中进行统一管理和配置。



2.3.2 maven工程的继承

在Java语言中，类之间是可以继承的，通过继承，子类就可以引用父类中非private的属性和方法。同样，在maven工程之间也可以继承，子工程继承父工程后，就可以使用在父工程中引入的依赖。继承的目的是为了消除重复代码。

在Java语言中，类之间是可以继承的，通过继承，子类就可以引用父类中非private的属性和方法。同样，在maven工程之间也可以继承，子工程继承父工程后，就可以使用在父工程中引入的依赖。继承的目的是为了消除重复代码。

被继承的Maven项目中的POM的部分定义是

```
<groupId>com.company</groupId>
<artifactId>company-project-parent</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>pom</packaging>
```

被继承的maven工程通常称为父工程，父工程的打包方式必须为pom，所以我们区分某个maven工程是否为父工程就看这个工程的打包方式是否为pom

继承的Maven项目中的POM的关键部分就是

```
<parent>
<groupId>com.taotao</groupId>
<artifactId>company-project-parent</artifactId>
<version>0.0.1-SNAPSHOT</version>
</parent>
<artifactId>company-project-children</artifactId>
```

继承其他maven父工程的工程通常称为子工程，在pom.xml文件中通过parent标签进行父工程的继承

2.3.3 maven工程的聚合

在maven工程的pom.xml文件中可以使用标签将其他maven工程聚合到一起，聚合的目的是为了进行统一操作。

例如拆分后的maven工程有多个，如果要进行打包，就需要针对每个工程分别执行打包命令，操作起来非常繁琐。这时就可以使用标签将这些工程统一聚合到maven父工程中，需要打包的时候，只需要在此工程中执行一次打包命令，其下被聚合的工程就都会被打包了。

```
<groupId>com.lagou</groupId>
<artifactId>lagou_edu_home_parent</artifactId>
<version>1.0-SNAPSHOT</version>
<modules>
  <module>ssm_domain</module>
  <module>ssm_utils</module>
  <module>ssm_dao</module>
  <module>ssm_service</module>
  <module>ssm_web</module>
</modules>
```

五个工程被聚合到一个工程

2.3.3 maven聚合工程_搭建拉勾教育后台管理系统

工程整体结构如下：

- 1) lagou_edu_home_parent为父工程，其余工程为子工程，都继承父工程lagou_edu_home_parent
- 2) lagou_edu_home_parent工程将其子工程都进行了聚合
- 3) 子工程之间存在依赖关系：

ssm_domain依赖ssm_utils

ssm_dao依赖ssm_domain

ssm_service依赖ssm_dao

① 父工程lagou_edu_home_parent构建

修改pom.xml,添加依赖

```

<properties>
    <spring.version>5.1.5.RELEASE</spring.version>
    <springmvc.version>5.1.5.RELEASE</springmvc.version>
    <mybatis.version>3.5.1</mybatis.version>
</properties>

<!-- 锁定jar版本 -->
<dependencyManagement>
    <dependencies>
        <!-- Mybatis -->
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis</artifactId>
            <version>${mybatis.version}</version>
        </dependency>
        <!-- springMVC -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>${springmvc.version}</version>
        </dependency>
        <!-- spring -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-aop</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-web</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-expression</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>

```

```

        <artifactId>spring-beans</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-aspects</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context-support</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-test</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-jdbc</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-tx</artifactId>
        <version>${spring.version}</version>
    </dependency>
</dependencies>
</dependencyManagement>

<dependencies>
    <!--mybatis坐标-->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>5.1.47</version>
    </dependency>
    <dependency>
        <groupId>com.alibaba</groupId>
        <artifactId>druid</artifactId>
        <version>1.1.15</version>
    </dependency>
    <dependency>
        <groupId>org.mybatis</groupId>
        <artifactId>mybatis</artifactId>
    </dependency>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
    </dependency>

    <!--spring坐标-->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
    </dependency>

```

```
<dependency>
    <groupId>org.aspectj</groupId>
    <artifactId>aspectjweaver</artifactId>
    <version>1.8.13</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-tx</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-test</artifactId>
</dependency>

<!--mybatis整合spring坐标-->
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis-spring</artifactId>
    <version>1.3.1</version>
</dependency>

<!--springMVC坐标-->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
</dependency>
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>3.1.0</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>javax.servlet.jsp</groupId>
    <artifactId>jsp-api</artifactId>
    <version>2.2</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>jstl</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
</dependency>

<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.9.8</version>
</dependency>
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-core</artifactId>
    <version>2.9.8</version>
```

```

</dependency>
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-annotations</artifactId>
    <version>2.9.0</version>
</dependency>

<dependency>
    <groupId>com.github.pagehelper</groupId>
    <artifactId>pagehelper</artifactId>
    <version>4.1.6</version>
</dependency>

<!-- Beanutils -->
<dependency>
    <groupId>commons-beanutils</groupId>
    <artifactId>commons-beanutils</artifactId>
    <version>1.8.3</version>
</dependency>

<dependency>
    <groupId>commons-fileupload</groupId>
    <artifactId>commons-fileupload</artifactId>
    <version>1.3.1</version>
</dependency>

<!-- 解决跨域问题所需依赖 -->
<dependency>
    <groupId>com.thetransactioncompany</groupId>
    <artifactId>cors-filter</artifactId>
    <version>2.5</version>
</dependency>

</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.1</version>
            <configuration>
                <source>1.8</source>
                <target>1.8</target>
                <encoding>UTF-8</encoding>
            </configuration>
        </plugin>
    </plugins>
</build>

```

② 子工程ssm_utils构建

③ 子工程ssm_domain构建

```

<!-- 依赖ssm_utils -->
<dependencies>
    <dependency>
        <groupId>com.lagou</groupId>
        <artifactId>ssm_utils</artifactId>
        <version>1.0-SNAPSHOT</version>
    </dependency>
</dependencies>

```

④ 子工程ssm_dao构建

配置ssm_domain工程的pom.xml文件

```

<!--依赖ssm_domain-->
<dependencies>
    <dependency>
        <groupId>com.lagou</groupId>
        <artifactId>ssm_domain</artifactId>
        <version>1.0-SNAPSHOT</version>
    </dependency>
</dependencies>

```

创建DAO接口和Mapper映射文件

```

package com.lagou.dao;

import com.lagou.domain.Test;

import java.util.List;

public interface TestMapper {

    public List<Test> findAllTest();

}

```

```

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.lagou.dao.TestMapper">
    <select id="findAllTest" resultType="com.lagou.domain.Test">
        select * from test;
    </select>
</mapper>

```

在resources目录下创建spring配置文件applicationContext-dao.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:context="http://www.springframework.org/schema/context"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xmlns:aop="http://www.springframework.org/schema/aop"
    xsi:schemaLocation="

```

```

    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd
    http://www.springframework.org/schema/tx
    http://www.springframework.org/schema/tx/spring-tx.xsd
    http://www.springframework.org/schema/aop
    http://www.springframework.org/schema/aop/spring-aop.xsd">

    <!--spring整合mybatis-->
    <context:property-placeholder location="classpath:jdbc.properties">
</context:property-placeholder>

    <bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource">
        <property name="driverClassName" value="${jdbc.driver}"/>
        <property name="url" value="${jdbc.url}"/>
        <property name="username" value="${jdbc.username}"/>
        <property name="password" value="${jdbc.password}"/>
    </bean>

    <!--sqlSessionFactory的创建权交给了spring 生产sqlSession-->
    <bean id="sqlSessionFactory"
class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="dataSource"></property>
        <property name="typeAliasesPackage" value="com.lagou.domain"></property>

        <!--引入加载mybatis的核心配置文件，可以不用去加载-->
        <property name="configLocation" value="classpath:SqlMapConfig.xml">
</property>
    </bean>

    <!--mapper映射扫描 MapperScannerConfigurer扫描该包下所有接口，生成代理对象存到IOC容
器中-->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <property name="basePackage" value="com.lagou.dao"></property>
    </bean>

</beans>

```

sqlMapConfig.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>
        <!-- 是否开启自动驼峰命名规则（camel case）映射，即从数据库列名 A_COLUMN 到属性名
aColumn 的类似映射 a_name aName-->
        <setting name="mapUnderscoreToCamelCase" value="true"/>
    </settings>
</configuration>

```

⑤子工程ssm_service构建

第一步：创建ssm_service工程

第二步：配置ssm_service工程的pom.xml文件

```
<dependencies>
    <dependency>
        <groupId>com.lagou</groupId>
        <artifactId>ssm_dao</artifactId>
        <version>1.0-SNAPSHOT</version>
    </dependency>
</dependencies>
```

第三步：创建TestService接口和实现类

```
public interface TestService {

    public List<Test> findAllTest();

}
```

```
@Service
public class TestServiceImpl implements TestService {

    @Autowired
    private TestMapper testMapper;

    @Override
    public List<Test> findAllTest() {
        List<Test> allTest = testMapper.findAllTest();
        return allTest;
    }
}
```

第四步：创建spring配置文件applicationContext-service.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:context="http://www.springframework.org/schema/context"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xmlns:aop="http://www.springframework.org/schema/aop"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context.xsd
        http://www.springframework.org/schema/tx
        http://www.springframework.org/schema/tx/spring-tx.xsd
        http://www.springframework.org/schema/aop
        http://www.springframework.org/schema/aop/spring-aop.xsd">

    <!--配置IOC相关操作:开启注解扫描-->
```



```

        <context:component-scan base-package="com.lagou.service">
</context:component-scan>

        <import resource="classpath:applicationContext_dao.xml"/>

</beans>

```

⑥子工程ssm_web构建

第一步：创建maven_web工程，注意打包方式为war

第二步：配置maven_web工程的pom.xml文件

```

<dependencies>
    <dependency>
        <groupId>com.lagou</groupId>
        <artifactId>ssm_service</artifactId>
        <version>1.0-SNAPSHOT</version>
    </dependency>
</dependencies>

```

第三步：创建Controller

```

@RestController
@RequestMapping("/test")
public class TestController {

    @Autowired
    private TestService testService;

    @RequestMapping("/findAllTest")
    public List<Test> findAllTest(){
        List<Test> allTest = testService.findAllTest();
        return allTest;
    }

}

```

第四步：创建springmvc配置文件springmvc.xml

```

<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">

    <!--1.组件扫描：只扫描controller-->

```

```

<context:component-scan base-package="com.lagou.controller">
</context:component-scan>

<!--2.mvc注解增强:处理器映射器及处理器适配器-->
<mvc:annotation-driven></mvc:annotation-driven>

<!--3.视图解析器-->
<!--    <bean id="resourceViewResolve"
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/"></property>
        <property name="suffix" value=".html"></property>
    </bean>-->

<!--4.放行静态资源-->
<mvc:default-servlet-handler></mvc:default-servlet-handler>

</beans>

```

第五步：编写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:context="http://www.springframework.org/schema/context"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xmlns:aop="http://www.springframework.org/schema/aop"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context.xsd
        http://www.springframework.org/schema/tx
        http://www.springframework.org/schema/tx/spring-tx.xsd
        http://www.springframework.org/schema/aop
        http://www.springframework.org/schema/aop/spring-aop.xsd">

    <!--引入: applicationContext_service.xml-->
    <import resource="classpath:applicationContext_service.xml"/>

</beans>

```

第六步：配置web.xml

```

<!--前端控制器-->
<servlet>
    <servlet-name>DispatcherServlet</servlet-name>
    <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
        <param-name>contextConfigLocation</param-name>

```

```

        <param-value>classpath:spring-mvc.xml</param-value>
    </init-param>
    <load-on-startup>2</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>DispatcherServlet</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>

<!--中文乱码过滤器：解决post方式提交的乱码-->
<filter>
    <filter-name>CharacterEncodingFilter</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>CharacterEncodingFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>

<!--配置spring的监听器-->
<listener>
    <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>

<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:applicationContext.xml</param-value>
</context-param>

<!--配置跨域过滤器-->
<filter>
    <filter-name>corsFilter</filter-name>
    <filter-class>com.thetransactioncompany.cors.CORSFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>corsFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>

```

3.拉勾教育后台管理系统研发

3.1 课程管理模块功能分析

在本次的项目中,首先先来完成拉勾教育后台管理系统的 课程管理模块, 课程管理模块包含了多条件查询、 图片上传、 新建&修改课程、 课程状态管理

、课程内容展示、回显章节对应的课程信息、新建&修改章节信息、修改章节状态、新建&修改课时信息等接口的编写

3.1.1 课程管理

- 实现以下功能:
 - 多条件查询
 - 图片上传
 - 新建课程信息
 - 回显课程信息
 - 修改课程信息
 - 课程状态管理
 - 课程内容展示
 - 回显章节对应的课程信息
 - 新建&修改章节信息
 - 修改章节状态
 - 新建课时信息

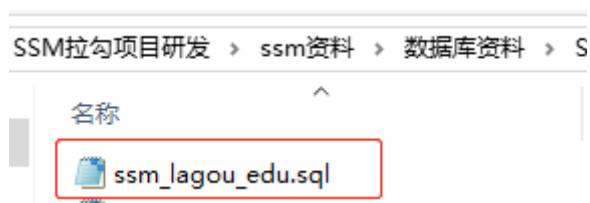


ID	课程名称	价格	排序	状态	操作
11	大数据	¥199	20	●	<button>下架</button> <button>编辑</button> <button>内容管理</button>
10	React 入门	¥164	12	●	<button>上架</button> <button>编辑</button> <button>内容管理</button>
8	Vue.js 3.0 核心源码解析	¥99	7	●	<button>下架</button> <button>编辑</button> <button>内容管理</button>
7	文案高手的18项修炼	¥263	1	●	<button>下架</button> <button>编辑</button> <button>内容管理</button>
9	秒杀11	¥200	1	●	<button>下架</button> <button>编辑</button> <button>内容管理</button>

3.2 课程管理模块表设计

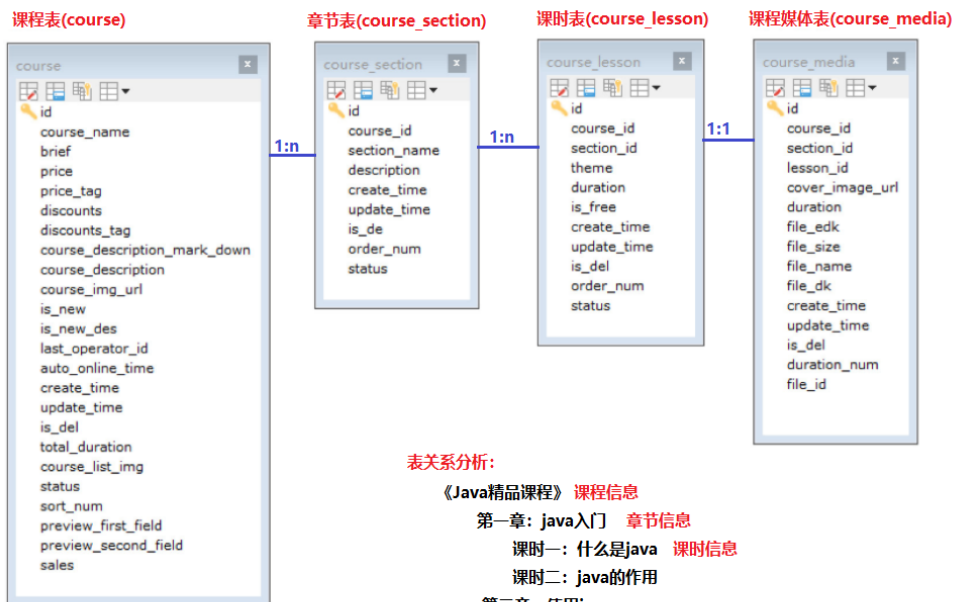
3.2.1 创建数据库及表

在资料中找到 ssm_lagou_edu.sql,使用SQLYog 执行SQL脚本 ,导入表结构及表信息



3.2.2 表关系介绍

1.ER图



2.数据实体描述

2.1 课程表

字段名称	类型	约束	描述
id	int(11) unsigned NOT NUL AUTO_INCREMENT	PK	主键PK
course_name	varchar(255)		课程名
brief	varchar(255)		课程一句话简介
price	double(10,2)		原价
price_tag	varchar(255)		原价标签
discounts	double(10,2)		优惠价
discounts_tag	varchar(255)		优惠标签
course_description_mark_down	longtext		描述markdown
course_description	longtext		课程描述
course_img_url	varchar(255)		课程列表图
is_new	tinyint(1)		是否新品
is_new_des	varchar(255)		广告语
last_operator_id	int(11)		最后操作者
create_time	datetime		创建时间
update_time	datetime		更新时间'
is_del	tinyint(1)		是否删除
total_duration	int(11)		总时长(分钟)
course_list_img	varchar(255)		课程列表展示图片
status	int(2)		课程状态, 0-草稿, 1-上架
sort_num	int(11)		课程排序, 用于后台保存草稿时用到
preview_first_field	varchar(255)		课程预览第一个字段
preview_second_field	varchar(255)		课程预览第二个字段
sales	int(11)		销量

2.2 章节表

字段名称	类型	约束	描述
id	int(11)	PK	'主键ID
course_id	int(11)		课程id
section_name	varchar(255)		章节名
description	varchar(255)		'章节描述'
status	int(1)		状态, 0:隐藏; 1: 待更新; 2: 已发布
order_num	int(11)		排序字段
is_de	tinyint(1)		是否删除
update_time	datetime		'更新时间
create_time	datetime		记录创建时间

2.3 课时表

字段名称	类型	约束	描述
id	int(11)	PK	主键
course_id	int(11)		课程id
section_id	int(11)		章节id
theme	varchar(255)		课时主题
duration	int(11)		'课时时长(分钟)
is_free	tinyint(1)		是否免费
is_del	tinyint(1)		是否删除
order_num	int(11)		排序字段
status	int(2)		课时状态,0-隐藏, 1-未发布, 2-已发布
update_time	datetime		更新时间
create_time	datetime		记录创建时间

2.4 课程媒体

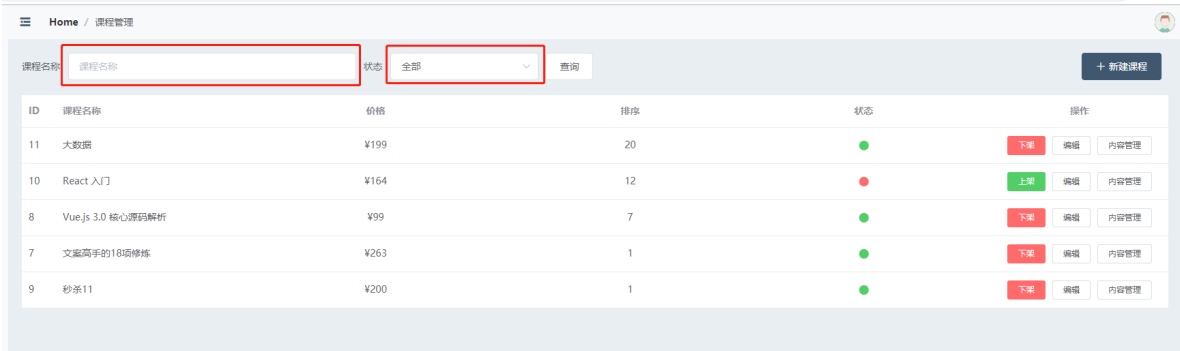
字段名称	类型	约束	描述
id	int(11)	PK	'课程媒体主键ID
course_id	int(11)		课程id
section_id	int(11)		章节id
cover_image_url	varchar(255)		封面图URL
duration	varchar(50)		时长 (06:02)
file_edk	varchar(500)		媒体资源文件对应的EDK
file_size	double(10,2)		文件大小MB
file_name	varchar(100)		文件名称
file_dk	varchar(100)		'媒体资源文件对应的DK
duration_num	varchar(50)		时长, 秒数
file_id	varchar(50)		媒体资源文件ID
is_del	tinyint(1)		是否删除, 0未删除, 1删除
update_time	datetime		更新时间
create_time	datetime		创建时间

3.3 课程管理模块接口实现

多条件课程列表查询

3.1 需求分析

根据课程名称及课程状态进行多条件查询



3.2 查看接口文档，进行编码

查看接口文档

实体类：Course

```
//主键
```



```
private int id;

//课程名称
private String courseName;

//课程一句话简介
private String brief;

//原价
private double price;

//原价标签
private String priceTag;

//优惠价
private double discounts;

//优惠价标签
private String discountsTag;

//课程内容markdown
private String courseDescriptionMarkDown;

//课程描述
private String courseDescription;

//课程分享图片url
private String courseImgUrl;

//是否新品
private int isNew;

//广告语
private String isNewDes;

//最后操作者
private int lastOperatorId;

//自动上架时间
private Date autoOnlineTime;

//创建时间
private Date createTime;

//更新时间
private Date updateTime;

//是否删除
private int isDel;

//总时长
private int totalDuration;

//课程列表展示图片
private String courseListImg;

//课程状态，0-草稿，1-上架
private int status;
```

```
//课程排序
private int sortNum;

//课程预览第一个字段
private String previewFirstField;

//课程预览第二个字段
private String previewSecondField;

// getter/setter....
```

ResponseResult

```
public class ResponseResult {
    private Boolean success;
    private Integer state;
    private String message;
    private Object content;

    public ResponseResult() {
    }

    public ResponseResult(Boolean success, Integer state, String message, Object
content) {
        this.success = success;
        this.state = state;
        this.message = message;
        this.content = content;
    }

    //getter/setter..
}
```

实体类：CourseVo(View Object表现层对象：主要用于表现层来接收参数的)

```
public class CourseVO {

    /**
     * 课程名称
     */
    private String courseName;

    /**
     * 课程状态
     */
    private Integer status;

    // getter/setter....
}
```

Dao层：CourseMapper

```

public interface CourseMapper {

    /**
     * 多条件课程列表查询
     */
    public List<Course> findCourseByConditioin(CourseVo courseVo);

}

```

```

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.lagou.dao.CourseMapper">

    <!-- 多条件课程列表查询 -->
    <select id="findCourseByConditioin"
parameterType="com.lagou.domain.CourseVO" resultType="com.lagou.domain.Course">
        SELECT
            id,
            course_name,
            price,
            sort_num,
            STATUS
        FROM
            course
        <where>
            <if test="true">
                and is_del != 1
            </if>
            <if test="courseName != null and courseName != ''">
                and course_name like concat('%',#{courseName},'%')
            </if>
            <if test="status != null and status != ''">
                and status = #{status}
            </if>
        </where>

    </select>

```

Service层: CourseService

```

public interface CourseService {

    /**
     * 多条件查询课程列表
     */
    public PageInfo findCourseByConditioin(CourseVO courseVO);

}

```

```

@Service
public class CourseServiceImpl implements CourseService {

    @Autowired
    private CourseMapper courseMapper;

    @Override
    public List<Course> findCourseByConditioin(CourseVO courseVO) {

        List<Course> courseList = courseMapper.findCourseByConditioin(courseVO);

        return courseList;
    }
}

```

Web层: CourseController

```

@RestController
@RequestMapping("/course")
public class CourseController {

    @Autowired
    private CourseService courseService;

    /**
     * 查询课程信息&条件查询 接口
     * */
    @RequestMapping("/findCourseByConditioin")
    public ResponseResult findCourseByConditioin(@RequestBody CourseVO courseVO)
    {

        List<Course> courseList =
        courseService.findCourseByConditioin(courseVO);

        ResponseResult result = new ResponseResult(true,200,"响应成功",courseList);

        return result;
    }
}

```

Postman测试接口

课程图片上传

需求分析:

需求: 在新增课程页面需要进行图片上传, 并回显图片信息

← 返回

新增课程

保存

基本信息

* 名称

0/50

* 简介

0/100

* 讲师姓名

0/50

* 职位

0/50

* 讲师简介

0/100

* 课程概述

0/20

0/20

课程封面

点击上传

建议尺寸: 230*300px, JPG、PNG格式, 图片小于150K

解锁封面

点击上传

建议尺寸: 750*422px, JPG、PNG格式, 图片小于150K

查看接口文档，进行编码

查看接口文档

springmvc.xml

```
<!--5.配置文件解析器-->
<!-- 此处id为固定写法，不能随便取名-->
<bean id="multipartResolver"
class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
    <property name="maxUploadSize" value="1048576"></property>
</bean>
```

Web层: CourseController

```
/**
 * 图片上传接口
 * */
@RequestMapping("/courseUpload")
public ResponseResult fileUpload(@RequestParam("file")MultipartFile file,
HttpServletRequest request){

    try {
        //1.判断文件是否为空
        if(file.isEmpty()){
            throw new RuntimeException();
        }

        //2.获取项目部署路径
        // D:\apache-tomcat-8.5.56\webapps\ssm_web\
        String realPath = request.getServletContext().getRealPath("/");
        // D:\apache-tomcat-8.5.56\webapps\
        String webappsPath =
realPath.substring(0,realPath.indexOf("ssm_web"));
```

```

//3.获取原文件名
String fileName = file.getOriginalFilename();

//4.新文件名
String newFileName = System.currentTimeMillis() +
fileName.substring(fileName.lastIndexOf("."));

//5.上传文件
String uploadPath = webappsPath+"upload\\";
File filePath = new File(uploadPath,newFileName);

//如果目录不存在就创建目录
if(!filePath.getParentFile().exists()){
    filePath.getParentFile().mkdirs();
    System.out.println("创建目录: " + filePath);
}
file.transferTo(filePath);

//6.将文件名和文件路径返回
Map<String,String> map = new HashMap<>();
map.put("fileName",newFileName);
map.put("filePath",LOCAL_URL+"/upload/"+newFileName);
ResponseResult result = new ResponseResult(true,200,"响应成功",map);

return result;
} catch (Exception e) {
    e.printStackTrace();
    return null;
}
}

```

Postman测试接口

新建课程信息

需求分析：

填写好新增课程信息后，点击保存，将表单信息保存到数据库中

Home / 课程管理 / 新增课程

← 返回 新增课程 保存

基本信息

* 名称

0/50

* 简介

0/100

* 讲师姓名

0/50

* 职位

0/50

* 讲师简介

0/100

* 课程概述

0/20

0/20

课程封面

点击上传

建议尺寸：230*300px，JPG、PNG格式，图片小于150K

查看接口文档，进行编码

查看接口文档

Dao层：CourseMapper

```
public interface CourseMapper {  
  
    /**  
     * 保存课程信息  
     */  
    public int saveCourse(Course course);  
  
    /**  
     * 保存讲师信息  
     */  
    public void saveTeacher(Teacher teacher);  
  
}
```

CourseMapper.xml

```
<!-- 保存课程信息 -->  
<insert id="saveCourse" parameterType="com.lagou.domain.Course" >  
  
    INSERT INTO course(  
        course_name,  
        brief,  
        preview_first_field,  
        preview_second_field,  
        course_img_url,  
        course_list_img,  
        sort_num,  
        price,  
        discounts,  
        sales,  
        discounts_tag,  

```

```

        course_description_mark_down,
        create_time,
        update_time
    ) VALUES(#{courseName},#{brief},#{previewFirstField},#{previewSecondField},#{
{courseImgUrl}},
        #{courseListImg},#{sortNum},#{price},#{discounts},#{sales},#{discountsTag},#{
{courseDescriptionMarkDown}},
        #{createTime},#{updateTime});

<selectKey resultType="java.lang.Integer" order="AFTER" keyProperty="id">
    SELECT LAST_INSERT_ID();
</selectKey>
</insert>

<!-- 保存讲师信息 -->
<insert id="saveTeacher" parameterType="com.lagou.domain.Teacher">
    INSERT INTO teacher(
        course_id,
        teacher_name,
        POSITION,
        description,
        create_time,
        update_time
    ) VALUES(#{courseId},#{teacherName},#{position},#{description},#{
{createTime}},#{updateTime});
</insert>

```

Service层: CourseService

```

public interface CourseService {

    /**
     * 保存课程信息
     * */
    public void saveCourseOrTeacher(CourseVO courseVO);

}

```

```

@Service
public class CourseServiceImpl implements CourseService {

    @Autowired
    private CourseMapper courseMapper;

    @Override
    public void saveCourseOrTeacher(CourseVO courseVO) {

        try {
            //封装课程信息
            Course course = new Course();
            BeanUtils.copyProperties(course,courseVO);

            //补全信息

```



```

        Date date = new Date();
        course.setCreateTime(date);
        course.setUpdateTime(date);

        //保存课程
        courseMapper.saveCourse(course);

        //获取新插入数据的id
        int id = course.getId();

        //封装讲师信息
        Teacher teacher = new Teacher();
        BeanUtils.copyProperties(teacher, courseVO);

        //补全信息
        teacher.setCourseId(id);
        teacher.setCreateTime(date);
        teacher.setUpdateTime(date);

        //保存讲师信息
        courseMapper.saveTeacher(teacher);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

}

```

Web层: CourseController

```

@RestController
@RequestMapping("/course")
public class CourseController {

    @Autowired
    private CourseService courseService;

    /**
     * 保存&修改课程信息接口
     * */
    @RequestMapping("/saveOrUpdateCourse")
    public ResponseResult saveOrUpdateCourse(@RequestBody CourseVO courseVO){

        try {

            if(courseVO.getId() == null){
                courseService.saveCourseOrTeacher(courseVO);
                ResponseResult result = new ResponseResult(true,200,"响应成功",null);

                return result;
            }else{
                courseService.updateCourseOrTeacher(courseVO);
            }
        }
    }
}

```

Postman测试接口

回显课程信息

需求分析:

Home / 课程管理

课程名称

课程名称

状态

全部

查询

+ 新建课程

ID	课程名称	价格	排序	状态	操作
11	大数据	¥199	20	<div></div>	<div>下架</div> <div>编辑</div> <div>内容管理</div>
10	React 入门	¥164	12	<div></div>	<div>上架</div> <div>编辑</div> <div>内容管理</div>
8	Vue.js 3.0 核心源码解析	¥99	7	<div></div>	<div>下架</div> <div>编辑</div> <div>内容管理</div>
7	文案高手的18项修炼	¥263	1	<div></div>	<div>下架</div> <div>编辑</div> <div>内容管理</div>
9	秒杀11	¥200	1	<div></div>	<div>下架</div> <div>编辑</div> <div>内容管理</div>

查看接口文档，进行编码

[查看接口文档](#)

Dao层: CourseMapper

CourseMapper.xml

```

        POSITION,
        description,
        preview_first_field,
        preview_second_field,
        course_img_url,
        course_list_img,
        sort_num,
        discounts,
        price,
        sales,
        discounts_tag,
        course_description_mark_down
    FROM course LEFT JOIN teacher ON course.id = teacher.course_id
    WHERE course.id = #{id}
</select>

```

Service层: CourseService

```

public interface CourseService {

    /**
     * 根据id获取课程信息
     * */
    public CourseVO findCourseById(int id);

}

```

```

@Service
public class CourseServiceImpl implements CourseService {

    @Autowired
    private CourseMapper courseMapper;

    @Override
    public CourseVO findCourseById(int id) {
        return courseMapper.findCourseById(id);
    }

}

```

Web层: CourseController

```

@RestController
@RequestMapping("/course")
public class CourseController {

    @Autowired
    private CourseService courseService;

    /**
     * 根据id获取课程信息
     * */
}

```

```
@RequestMapping("/findCourseById")
public ResponseEntity findCourseById(@RequestParam int id) {
    try {
        CourseVO courseVO = courseService.findCourseById(id);
        ResponseEntity result = new ResponseEntity(true,200,"响应成功",courseVO);
        return result;

    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
}
```

Postman测试接口

修改课程信息

需求分析：

点击保存按钮，将修改后的课程信息保存到数据库中

Home / 课程管理 / 营销信息

返回

大数据

保存

基本信息

* 名称

大数据

3/50

* 简介

大数据

3/100

* 讲师姓名

大数据

3/50

* 职位

大数据

3/50

* 讲师简介

大数据

3/100

* 课程概述


大数据

3/20

大数据

3/20

课程封面



点击上传

建议尺寸：230*300px，JPG、PNG格式，图片小于150K

课程详情

H B I A * - 66 三 四 五 六 七 八 九 十 十一 十二 十三 十四 十五 十六 十七 十八 十九 二十 二十一 二十二 二十三 二十四 二十五 二十六 二十七 二十八 二十九 三十 三十一 三十二 三十三 三十四 三十五 三十六 三十七 三十八 三十九 四十 四十一 四十二 四十三 四十四 四十五 四十六 四十七 四十八 四十九 五十 五十一 五十二 五十三 五十四 五十五 五十六 五十七 五十八 五十九 六十 六十一 六十二 六十三 六十四 六十五 六十六 六十七 六十八 六十九 七十 七十一 七十二 七十三 七十四 七十五 七十六 七十七 七十八 七十九 八十 八十一 八十二 八十三 八十四 八十五 八十六 八十七 八十八 八十九 九十 九十一 九十二 九十三 九十四 九十五 九十六 九十七 九十八 九十九 一百

大数据大数据大数据大数据

大数据大数据大数据大数据

保存

查看接口文档，进行编码

查看接口文档

Dao层：CourseMapper

```
public interface CourseMapper {

    /**
     * 修改课程信息
     * */
    public void updateCourse(Course course);

    /**
     * 修改讲师信息
     * */
    public void updateTeacher(Teacher teacher);

}
```

CourseMapper.xml

```
<!-- 修改课程信息 -->
<update id="updateCourse" parameterType="com.lagou.domain.Course">
    UPDATE course
    <trim prefix="SET" suffixOverrides=",">
        <if test="courseName != null and courseName != ''">
            course_name = #{courseName},
        </if>

        <if test="brief != null and brief != ''">
            brief=#{brief},
        </if>

        <if test="previewFirstField != null and previewFirstField != ''">
            preview_first_field=#{previewFirstField},
        </if>

        <if test="previewSecondField != null and previewSecondField != ''">
            preview_second_field=#{previewSecondField},
        </if>

        <if test="courseImgUrl != null and courseImgUrl != ''">
            course_img_url=#{courseImgUrl},
        </if>

        <if test="courseListImg != null and courseListImg != ''">
            course_list_img=#{courseListImg},
        </if>

        <if test="sortNum != null and sortNum != ''">
            sort_num=#{sortNum},
        </if>
    </trim>
</update>
```

```

        </if>

        <if test="price != null and price != ''">
            price=#{price},
        </if>

        <if test="discounts != null and discounts != ''">
            discounts=#{discounts},
        </if>

        <if test="sales != null and sales != '' or sales==0">
            sales=#{sales},
        </if>

        <if test="discountsTag != null and discountsTag != ''">
            discounts_tag=#{discountsTag},
        </if>

        <if test="courseDescriptionMarkDown != null and
courseDescriptionMarkDown != ''">
            course_description_mark_down=#{courseDescriptionMarkDown},
        </if>

        <if test="updateTime != null">
            update_time=#{updateTime},
        </if>

    </trim>

    <where>
        <if test="id!=null and id != '' ">id=#{id}</if>
    </where>
</update>

<!-- 修改讲师信息 -->
<update id="updateTeacher" parameterType="com.lagou.domain.Teacher">
    UPDATE teacher
    <trim prefix="SET" suffixOverrides=",">
        <if test="teacherName != null and teacherName != ''">
            teacher_name = #{teacherName},
        </if>

        <if test="position != null and position != ''">
            position = #{position},
        </if>

        <if test="description != null and description != ''">
            description = #{description},
        </if>

        <if test="updateTime != null">
            update_time=#{updateTime}
        </if>
    </trim>
    <where>
        <if test="courseId != null and courseId != '' ">course_id = #
{courseId}</if>
    </where>

```

</update>

Service层: CourseService

```
public interface CourseService {  
  
    /**  
     * 修改课程信息  
     * */  
    public void updateCourseOrTeacher(CourseVO courseVO);  
  
}
```

```
@Service  
public class CourseServiceImpl implements CourseService {  
  
    @Autowired  
    private CourseMapper courseMapper;  
  
    @Override  
    public void updateCourseOrTeacher(CourseVO courseVO) {  
        try {  
            //封装课程信息  
            Course course = new Course();  
            BeanUtils.copyProperties(course, courseVO);  
  
            //补全信息  
            Date date = new Date();  
            course.setUpdateTime(date);  
  
            //更新课程  
            courseMapper.updateCourse(course);  
  
            //封装讲师信息  
            Teacher teacher = new Teacher();  
            BeanUtils.copyProperties(teacher, courseVO);  
  
            //补全信息  
            teacher.setCourseId(course.getId());  
            teacher.setUpdateTime(date);  
  
            //更新讲师信息  
            courseMapper.updateTeacher(teacher);  
        } catch (Exception e) {  
            e.printStackTrace();  
        }  
    }  
}
```

Web层: CourseController

```
@RestController  
@RequestMapping("/course")
```

```

public class CourseController {

    @Autowired
    private CourseService courseService;

    /**
     * 保存&修改课程信息接口
     * */
    @RequestMapping("/saveOrUpdateCourse")
    public ResponseEntity saveOrUpdateCourse(@RequestBody CourseVO courseVO){

        try {

            if(courseVO.getId() == null){
                courseService.saveCourseOrTeacher(courseVO);
                ResponseEntity result = new ResponseEntity(true,200,"响应成功",null);

                return result;

            }else{
                courseService.updateCourseOrTeacher(courseVO);
                ResponseEntity result = new ResponseEntity(true,200,"响应成功",null);

                return result;

            }

        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }

    }

}

```

Postman测试接口

课程状态管理

需求分析：

在课程列表展示页面中，可以通过点击 上架/下架按钮，修改课程状态

ID	课程名称	价格	排序	状态	操作
11	大数据	¥199	20	●	下架 编辑 内容管理
10	React 入门	¥164	12	●	上架 编辑 内容管理
8	Vue.js 3.0 核心源码解析	¥99	7	●	下架 编辑 内容管理
7	文案高手的18项修炼	¥263	1	●	下架 编辑 内容管理
9	秒杀11	¥200	1	●	下架 编辑 内容管理

查看接口文档，进行编码

查看接口文档

Dao层: CourseMapper

```
public interface CourseMapper {  
  
    /**  
     * 修改课程状态  
     * */  
    public void updateCourseStatus(Course course);  
}
```

CourseMapper.xml

```
<!-- 修改课程状态 -->  
<update id="updateCourseStatus" parameterType="com.lagou.domain.Course">  
    UPDATE course SET STATUS = #{status} ,update_time = #{updateTime} WHERE  
    id = #{id}  
</update>
```

Service层: CourseService

```
/**  
 * 修改课程状态  
 * */  
public void updateCourseStatus(int id,int status);
```

```
@Override  
public void updateCourseStatus(int id,int status) {  
  
    try {  
        //封装数据  
        Course course = new Course();  
        course.setStatus(status);  
        course.setId(id);  
        course.setUpdateTime(new Date());  
  
        //调用Dao  
        courseMapper.updateCourseStatus(course);  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

Web层: CourseController

```
/**  
 * 修改课程状态  
 * */  
@RequestMapping("/updateCourseStatus")
```

```
public ResponseResult updateCourseStatus(@RequestParam int id,@RequestParam
int status){

    try {
        //执行修改操作
        courseService.updateCourseStatus(id, status);

        //保存修改后的状态,并返回
        Map<String,Integer> map = new HashMap<>();
        map.put("status",status);
        ResponseResult result = new ResponseResult(true,200,"响应成功",map);

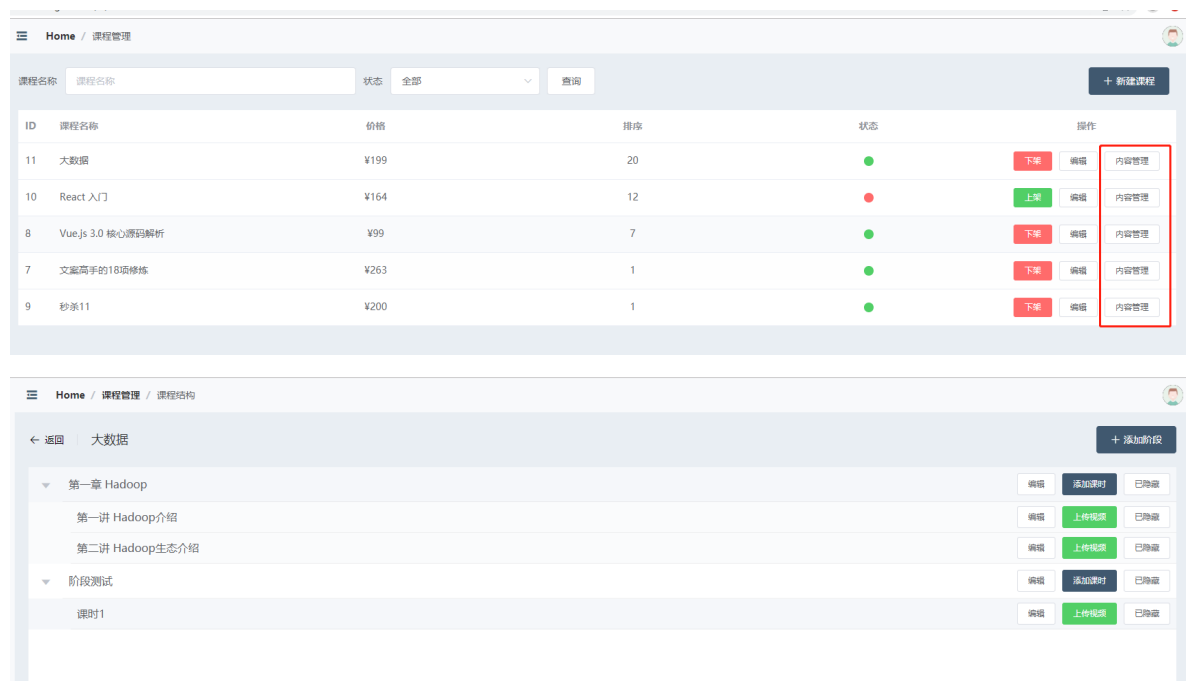
        return result;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return null;
}
```

Postman测试接口

课程内容展示

需求分析：

需求：点击内容管理，展示课程对应的课程内容（课程内容包含了章节和课时）



查看接口文档，进行编码：

查看接口文档：

CourseSection

```

/**
 * 章节类
 */
public class CourseSection {

    //id
    private Integer id;

    //课程id
    private int courseId;

    //章节名
    private String sectionName;

    //章节描述
    private String description;

    //创建时间
    private Date createTime;

    //更新时间
    private Date updateTime;

    //是否删除
    private int isDe;

    //排序
    private int orderNum;

    //状态
    private int status;

    //课时集合
    private List<CourseLesson> lessonList;

    public List<CourseLesson> getLessonList() {
        return lessonList;
    }

    public void setLessonList(List<CourseLesson> lessonList) {
        this.lessonList = lessonList;
    }
}

```

Dao层: CourseContentMapper

```

public interface CourseContentMapper {

    /**
     * 查询课程下的章节与课时信息
     */
    public List<CourseSection> findSectionAndLessonByCourseId(int courseId);

}

```

<!-- 根据课程ID 查询课程内容(章节与课时) -->

```

<select id="findSectionAndLessonByCourseId" parameterType="int"
resultMap="BaseResultMap">
    SELECT
    CS.*,
    <include refid="lesson_column_list"/>
    FROM course_section cs
    LEFT JOIN course_lesson cl ON cs.id = cl.section_id
    WHERE cs.course_id = #{courseId} ORDER BY cs.order_num ;
</select>

<!-- 一对多配置,一个章节下有多个课时 -->
<resultMap id="BaseResultMap" type="com.lagou.domain.CourseSection">
    <result property="id" column="id"></result>
    <result property="courseId" column="course_id"></result>
    <result property="sectionName" column="section_name"></result>
    <result property="description" column="description"></result>
    <result property="orderNum" column="order_num"></result>
    <result property="status" column="status"></result>

    <!-- 使用 collection,配置一对多关系 -->
    <collection property="lessonList" resultMap="lessonListResultMap"/>
</resultMap>

<resultMap id="lessonListResultMap" type="com.lagou.domain.CourseLesson">
    <id property="id" column="lessonId"></id>
    <result property="courseId" column="course_id"></result>
    <result property="sectionId" column="section_id"></result>
    <result property="theme" column="theme"></result>
    <result property="duration" column="duration"></result>
    <result property="isFree" column="is_free"></result>
    <result property="orderNum" column="order_num"></result>
    <result property="status" column="status"></result>
</resultMap>

<!-- 课时表字段信息 -->
<sql id="lesson_column_list">
    cl.id as lessonId,
    cl.course_id,
    cl.section_id,
    cl.theme,
    cl.duration,
    cl.is_free,
    cl.order_num,
    cl.status
</sql>

```

Service层: CourseContentService

```

public interface CourseContentService {

    public List<CourseSection> findSectionAndLessonByCourseId(int courseId);

}

```

```

@Service
public class CourseContentServiceImpl implements CourseContentService {

    @Autowired
    private CourseContentMapper contentMapper;

    @Override
    public List<CourseSection> findSectionAndLessonByCourseId(int courseId) {

        List<CourseSection> sectionList =
contentMapper.findSectionAndLessonByCourseId(courseId);
        return sectionList;
    }

}

```

Web层: CourseContentController

```

@RestController
@RequestMapping("/courseContent")
public class CourseContentController {

    @Autowired
    private CourseContentService contentService;

    /**
     * 查询课程内容
     * */
    @RequestMapping("/findSectionAndLesson")
    public ResponseEntity findSectionAndLessonByCourseId(@RequestParam int
courseId){

        try {
            //调用service
            List<CourseSection> sectionList =
contentService.findSectionAndLessonByCourseId(courseId);

            //封装数据并返回
            ResponseEntity result = new ResponseEntity(true,200,"响应成
功",sectionList);
            return result;

        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }

}

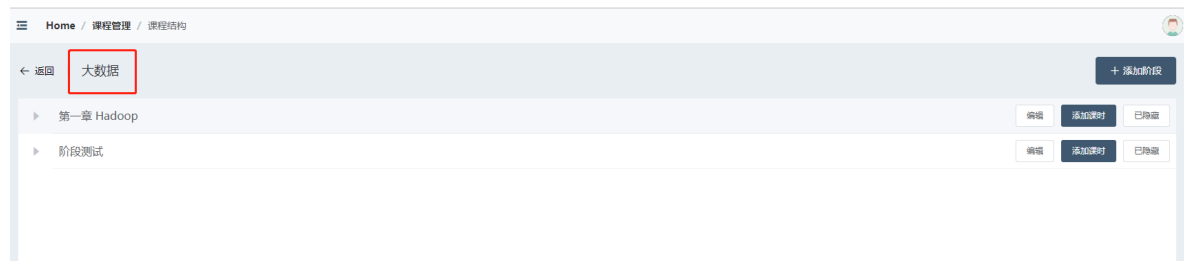
```

Postman测试接口

回显章节对应的课程信息

需求分析：

需求：在课程内容界面回显课程信息



查看接口文档，进行编码：

查看接口文档：

Dao层：CourseContentMapper

```
public interface CourseContentMapper {

    /**
     * 回显章节对应的课程信息
     * */
    public Course findCourseByCourseId(int courseId);

}
```

```
<!-- 回显课程信息 -->
<select id="findCourseByCourseId" parameterType="int"
resultType="com.lagou.domain.Course">
    SELECT id,course_name FROM course WHERE id = #{courseId}
</select>
```

Service层：CourseContentService

```
public interface CourseContentService {

    public Course findCourseByCourseId(int courseId);

}
```

```
@Service
public class CourseContentServiceImpl implements CourseContentService {

    @Autowired
    private CourseContentMapper contentMapper;
```

```

@Override
public Course findCourseByCourseId(int courseId) {

    Course course = contentMapper.findCourseByCourseId(courseId);
    return course;
}
}

```

Web层: CourseContentController

```

@RestController
@RequestMapping("/courseContent")
public class CourseContentController {

    @Autowired
    private CourseContentService contentService;

    /**
     * 回显章节对应的课程信息
     * */
    @RequestMapping("/findCourseByCourseId")
    public ResponseEntity findCourseByCourseId(@RequestParam int courseId){

        try {
            //调用service
            Course course = contentService.findCourseByCourseId(courseId);
            ResponseEntity result = new ResponseEntity(true,200,"响应成功",course);

            return result;
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }
}

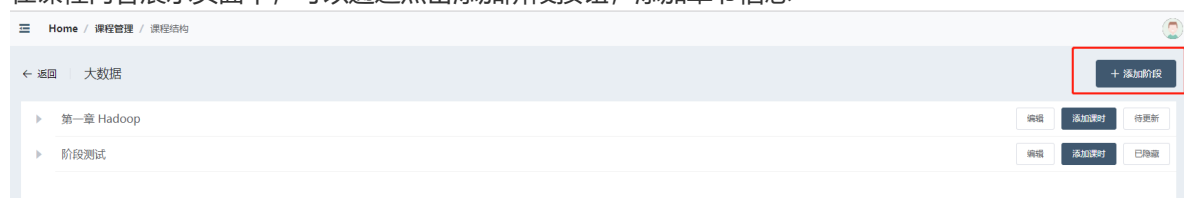
```

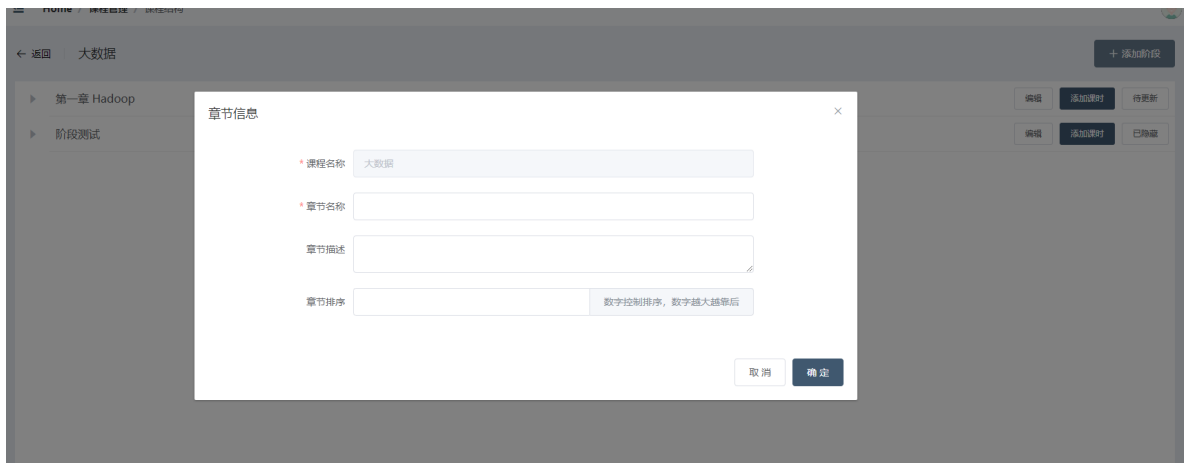
Postman测试接口

新建章节信息

需求分析:

在课程内容展示页面中, 可以通过点击添加阶段按钮, 添加章节信息





查看接口文档，进行编码

查看接口文档

Dao层：CourseContentMapper

```
/**
 * 保存章节
 * */
public void saveSection(CourseSection section);
```

```
<!-- 保存章节 -->
<insert id="saveSection" parameterType="com.lagou.domain.CourseSection">
    INSERT INTO course_section(
        course_id,
        section_name,
        description,
        order_num,
        STATUS,
        create_time,
        update_time
    )VALUES(#{courseId},#{sectionName},#{description},#{orderNum},
        #{status},#{createTime},#{updateTime});
</insert>
```

Service层：CourseContentService

```
public void saveSection(CourseSection section);
```



```

@Override
public void saveSection(CourseSection section) {

    //补全信息
    Date date = new Date();
    section.setCreateTime(date);
    section.setUpdateTime(date);

    contentMapper.saveSection(section);
}

```

Web层: CourseContentController

```

@RequestMapping("/saveOrUpdateSection")
public ResponseEntity saveOrUpdateSection(@RequestBody CourseSection
section) {

    try {

        contentService.saveSection(section);
        return new ResponseEntity(true, 200, "响应成功", null);
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
}

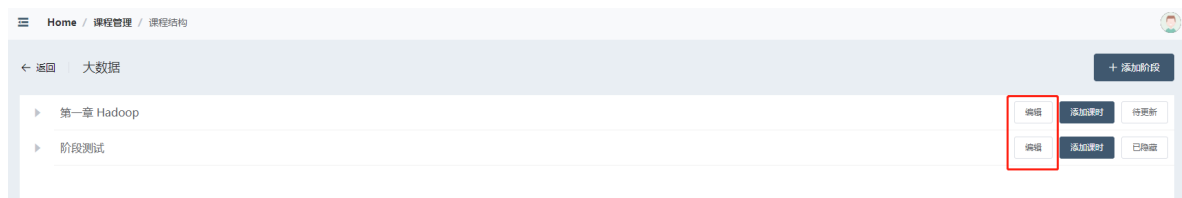
```

Postman测试接口

修改章节信息

需求分析:

点击确定按钮，将修改后的章节信息保存到数据库中



课时基本信息

*

课程名称

大数据

*

章节名称

第一章 Hadoop

*

课时名称

*

时长分钟

*

是否开放试听☐

课时排序

0

数字控制排序，数字越大越靠后

取消

确定

查看接口文档，进行编码

查看接口文档

Dao层：CourseContentMapper

```
/**
 * 修改章节
 * */
public void updateSection(CourseSection section);
```

```
<!-- 修改章节 -->
<update id="updateSection" parameterType="com.lagou.domain.CourseSection">
    UPDATE course_section
    <trim prefix="SET" suffixOverrides=",">
        <if test="sectionName != null and sectionName != ''">
            section_name = #{sectionName},
        </if>

        <if test="description != null and description != ''">
            description = #{description},
        </if>

        <if test="orderNum != null and orderNum != '' or orderNum == 0">
            order_num = #{orderNum},
        </if>

        <if test="updateTime != null">
            update_time=#{updateTime}
        </if>
    </trim>
    <where>
        <if test="id != null and id != '' ">id = #{id}</if>
```

```
</where>
</update>
```

Service层: CourseContentService

```
public void updateSection(CourseSection section);
```

```
@Override
public void updateSection(CourseSection section) {

    //补全信息
    Date date = new Date();
    section.setUpdateTime(date);

    contentMapper.updateSection(section);
}
```

Web层: CourseContentController

```
/**
 * 保存&修改章节信息
 */
@RequestMapping("/saveOrUpdateSection")
public ResponseResult saveOrUpdateSection(@RequestBody CourseSection
section) {

    try {
        //判断携带id是修改操作否则是插入操作
        if(section.getId() == null){
            contentService.saveSection(section);
            return new ResponseResult(true,200,"响应成功",null);
        }else{
            contentService.updateSection(section);
            return new ResponseResult(true,200,"响应成功",null);
        }
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
}
```

Postman测试接口

修改章节状态

需求分析:

需求: 点击修改章节状态按钮, 将当前章节信息的状态进行修改



查看接口文档，进行编码

查看接口文档

Dao层: CourseContentMapper

```
/**
 * 修改章节状态
 * */
public void updateSectionStatus(CourseSection section);
```

CourseContentMapper.xml

```
<!-- 修改章节状态 -->
<update id="updateSectionStatus"
parameterType="com.lagou.domain.CourseSection">
    UPDATE course_section set
        status = #{status},
        update_time = #{updateTime}
    WHERE id = #{id}
</update>
```

Service层: CourseContentService

```
public void updateSectionStatus(int id,int status);
```

```

@Override
public void updateSectionStatus(int id,int status) {

    //封装数据
    CourseSection section = new CourseSection();
    section.setId(id);
    section.setStatus(status);
    section.setUpdateTime(new Date());

    contentMapper.updateSectionStatus(section);
}

```

Web层: CourseContentController

```

/**
 * 修改章节状态
 * 状态, 0:隐藏; 1: 待更新; 2: 已发布
 * */
@RequestMapping("/updateSectionStatus")
public ResponseResult updateSectionStatus(@RequestParam int id,@RequestParam
int status){

    try {
        contentService.updateSectionStatus(id,status);

        //封装最新的状态信息
        Map<String,Integer> map = new HashMap<>();
        map.put("status",status);

        ResponseResult result = new ResponseResult(true,200,"响应成功",map);
        return result;

    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
}

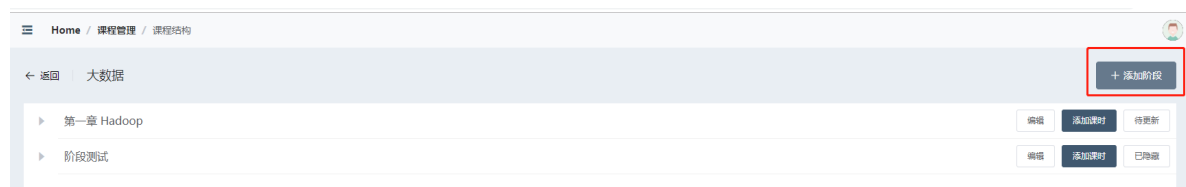
```

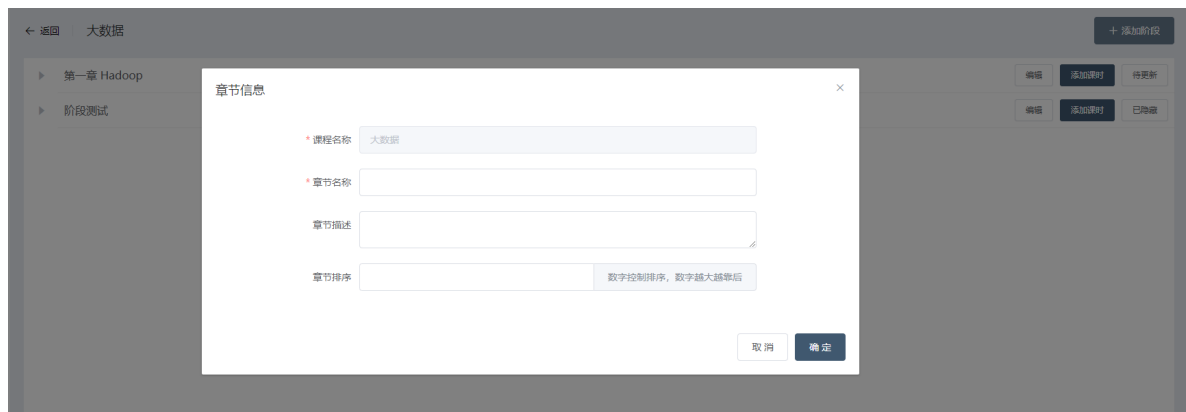
Postman测试接口

新建课时信息（自行完成）

需求分析：

需求：点击添加阶段按钮，将弹出页面填写的章节信息保存到数据库中





查看接口文档，进行编码

查看接口文档

Dao层: CourseContentMapper

```
/**
 * 保存课时
 * */
public void saveLesson(CourseLesson lesson);
```

```
<!-- 添加课时 -->
<insert id="saveLesson" parameterType="com.lagou.domain.CourseLesson">
    INSERT INTO course_lesson (
        id,course_id,
        section_id,
        theme,
        duration,
        is_free,
        order_num,
        create_time,
        update_time
    )VALUES(#{id},#{courseId},#{sectionId},#{theme},#{duration},#{isFree},
        #{orderNum},#{createTime},#{updateTime});
</insert>
```

Service层: CourseContentService

```
public void saveLesson(CourseLesson lesson);
```

```

@Override
public void saveLesson(CourseLesson lesson) {

    //补全信息
    Date date = new Date();
    lesson.setCreateTime(date);
    lesson.setUpdateTime(date);

    contentMapper.saveLesson(lesson);
}

```

Web层: CourseContentController

```

/**
 * 保存&修改课时
 * */
@RequestMapping("/saveOrUpdateLesson")
public ResponseEntity saveOrUpdateLesson(@RequestBody CourseLesson lesson){

    try {
        if(lesson.getId() == null){
            contentService.saveLesson(lesson);
            return new ResponseEntity(true,200,"响应成功",null);

        }else{
            contentService.updateLesson(lesson);
            return new ResponseEntity(true,200,"响应成功",null);
        }
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
}
}

```

Postman测试接口