任务二 课程管理模块开发1

1. 开发流程

1.1 需求分析

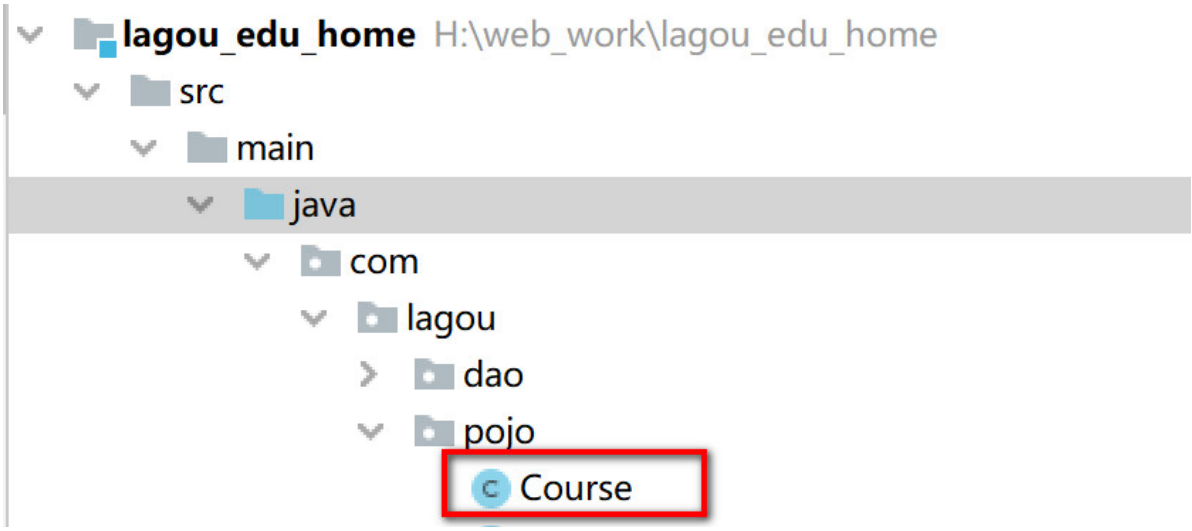
1.2 数据库表分析

这里展示的是我们需要使用的部分表字段

share_title 分享标题
share_description 分享描述
course_description 课程描述
sort_num 排序
status 课程状态,0-草稿,1-上架
create_time 创建时间
update_time 修改时间
is_del 是否删除 0-未删除 1-已删除

1.3 实体类设计

根据数据库中的Course表,对应创建 Course.java



1. 使用 @JSONField(ordinal = int类型的值)，指定排序的值,生成JSON时会按照指定顺序进行排序
2. 使用 @JSONField(serialize = false) ,排除不需要转换的字段,另外fastjson还会自动排除为空的字段

```
/**
 * 课程类
 * */
@Data
public class Course implements Serializable {

  //使用 JSONField 设置ordinal的值,来对转换成的JSON数据进行排序
```

```
    //课程ID
    @JSONField(ordinal = 1)
    private int id;

    //课程名称
    @JSONField(ordinal = 2)
    private String course_name;

    //课程介绍
    @JSONField(ordinal = 3)
    private String brief;

    //讲师名称
    @JSONField(ordinal = 4)
    private String teacher_name;

    //讲师介绍
    @JSONField(ordinal = 5)
    private String teacher_info;

    //课程原价
    @JSONField(ordinal = 6)
    private double price;

    //原价标签
    @JSONField(ordinal = 7)
    private String price_tag;

    //课程优惠价
    @JSONField(ordinal = 8)
    private double discounts;

    //课程概述
    @JSONField(ordinal = 9)
    private String preview_first_field;

    //课程概述第二个字段
    @JSONField(ordinal = 10)
    private String preview_second_field;

    //分享图片url
    @JSONField(ordinal = 11)
    private String course_img_url;

    //分享标题
    @JSONField(ordinal = 12)
    private String share_title;

    //分享描述
    @JSONField(ordinal = 13)
    private String share_description;

    //课程描述
    @JSONField(ordinal = 14)
    private String course_description;

    //排序
    @JSONField(ordinal = 15)
```

```
    private int sort_num;

    //课程状态,0-草稿,1-上架
    @JSONField(ordinal = 16)
    private int status;

    //创建时间
    @JSONField(ordinal = 17)
    private String create_time;

    //修改时间
    @JSONField(ordinal = 18)
    private String update_time;

    //是否删除
    @JSONField(ordinal = 19)
    private int isDel;

    @JSONField(ordinal = 20)
    private String share_image_title; //分享图title


    //使用JSONField(serialize = false)排除不需要转换的字段

    @JSONField(serialize = false)
    private int total_course_time; //课时数

    @JSONField(serialize = false)
    private int sales; //显示销量

    @JSONField(serialize = false)
    private int actual_sales; //真实销量

    @JSONField(serialize = false)
    private int is_new; //是否新品

    @JSONField(serialize = false)
    private String is_new_des; //广告语

    @JSONField(serialize = false)
    private int last_operator_id; //最后操作者

    @JSONField(serialize = false)
    private int total_duration; //总时长

    @JSONField(serialize = false)
    private long course_type; //课程类型

    @JSONField(serialize = false)
    private String last_notice_time;  //最后课程最近通知时间

    @JSONField(serialize = false)
    private long is_gray; //是否是灰度课程

    @JSONField(serialize = false)
    private long grade; //级别
```

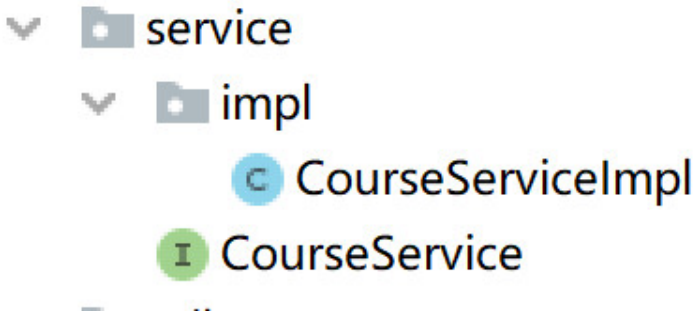
```
}
```

1.4 Dao接口及实现类编写


```
/**
 * 课程模块 DAO层接口
 * */
public interface CourseDao {

}
```

```
/**
 * 课程模块 DAO层实现类
 * */
public class CourseDaoImpl implements CourseDao {

}
```

1.5 Service接口及实现类编写




```
/**
 *  课程模块 Service层 接口
 * */
public interface CourseService {

}
```

```
/**
 * 课程模块Service层 实现类
 * */
public class CourseServiceImpl implements CourseService {

}
```

1.6 CourseServlet编写

CourseServlet 要继承通用的BaseServlet.

```
@WebServlet(name="courseServlet",value="/course")
public class CourseServlet extends BaseServlet {

}
```

2. 功能一: 查询课程列表信息

2.1 需求分析

页面分析,需要展示哪些数据

2.2 编写代码

2.2.1 Dao层编写

1. 修改CourseDao,添加 **findCourseList** 方法

```
接口 CourseDao
    //查询课程列表信息
    public List<Course> findCourseList();

实现类 CourseDaoImpl
     @Override
    public List<Course> findCourseList() {

        try {
            //1.创建QueryRunner
            QueryRunner qr = new QueryRunner(DruidUtils.getDataSource());
```

```
            //2.编写SQL
        String sql = "SELECT id,course_name,price,sort_num,STATUS FROM course
where id_del =  ?";

            //3.执行查询
          List<Course> courseList = qr.query(sql, new BeanListHandler<Course>
(Course.class), 0);

            return courseList;

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

逻辑删除

- 逻辑删除的本质是**修改操作**，所谓的逻辑删除其实并不是真正的删除，而是在表中将对应的是否删除标识,做修改操作。比如: 0是未删除，1是删除。在逻辑上数据是被删除的，但数据本身依然存在库中。

物理删除

- 物理删除就是真正的从数据库中做删除操作了。

2.2.2 Service层编写

修改CourseService 添加 **findCourseList** 方法

```
接口 CourseService
    public List<Course> findCourseList();

实现类 CourseServiceImpl
    //创建 CourseDao
    CourseDao courseDao = new CourseDaoImpl();

    @Override
    public List<Course> findCourseList() {

        //调用Dao 进行查询
        return courseDao.findCourseList();
    }
```

2.2.3 Servlet编写

2.2.3.1 接口开发规范

我们在做的是一个前后端分离项目、需要通过接口文档对接的项目. 所以开发过程中要仔细查看前端所需的api接口和参数字段

为了严格按照接口进行开发，提高效率，对请求及响应格式进行规范化。

开发规范
2、post请求时有三种数据格式，可以提交form表单数据 和 Json数据（ContentType=application/json），文件等多部件类型（multipart/form-data）三种数据格式 . jsonl类型的数据 Servlet中使用 fastjson进行解析
3、响应结果统一格式为json

开发规范
1、get 请求时，采用key/value格式请求，Servlet中可以使用 getParameter() 获取。
2、post请求时有三种数据格式 第一种: Json数据 ,jsonl类型的数据 Servlet中使用 fastjson进行解析 第二种: 提交form表单数据 第三种: 文件等多部件类型（multipart/form-data）
3、响应结果统一格式为json

为什么使用JSON?

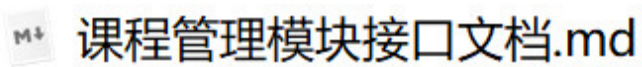
> 数据格式比较简单, 易于读写, JSON格式能够直接为服务器端代码使用, 大大简化了服务器端和客户端的代码开发量, 但是完成的任务不变, 且易于维护

本项目使用的是 JSON解析工具为阿里巴巴的fastjson, maven工程导入下面的依赖即可.

```
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>fastjson</artifactId>
    <version>1.1.37</version>
</dependency>

<dependency>
    <groupId>com.colobu</groupId>
    <artifactId>fastjson-jaxrs-json-provider</artifactId>
    <version>0.3.1</version>
</dependency>
```

2.2.3.2 接口文档

前端的开发基于服务端编写的接口，如果前端人员等待服务端人员将接口开发完毕再去开发前端内容这样做效率是 非常低下的，所以当接口定义完成，可以使用工具生成接口文档，前端人员查看接口文档即可进行前端开发，这样 前端和服务人员并行开发，大大提高了生产效率.



2.2.3.3 编写CourseServlet

在CourseServlet中添加 **findCourseList**方法

```
@WebServlet("/course")
public class CourseServlet extends BaseServlet {

    //查询课程信息列表
    public void findCourseList(HttpServletRequest request, HttpServletResponse
response){

        try {
            //1.接收参数

            //2.业务处理
            CourseService cs = new CourseServiceImpl();
            List<Course> courseList = cs.findCourseList();

            //3.响应结果
            //SimplePropertyPreFilter 指定要转换的JSON字段
            SimplePropertyPreFilter filter = new
SimplePropertyPreFilter(Course.class,
                    "id","course_name","price","sort_num","status");

            String result = JSON.toJSONString(courseList,filter);
            response.getWriter().print(result);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

2.3 Postman

2.3.1 postMan介绍

Postman是一款功能强大的http接口测试工具，使用postman可以完成http各种请求的功能测试。

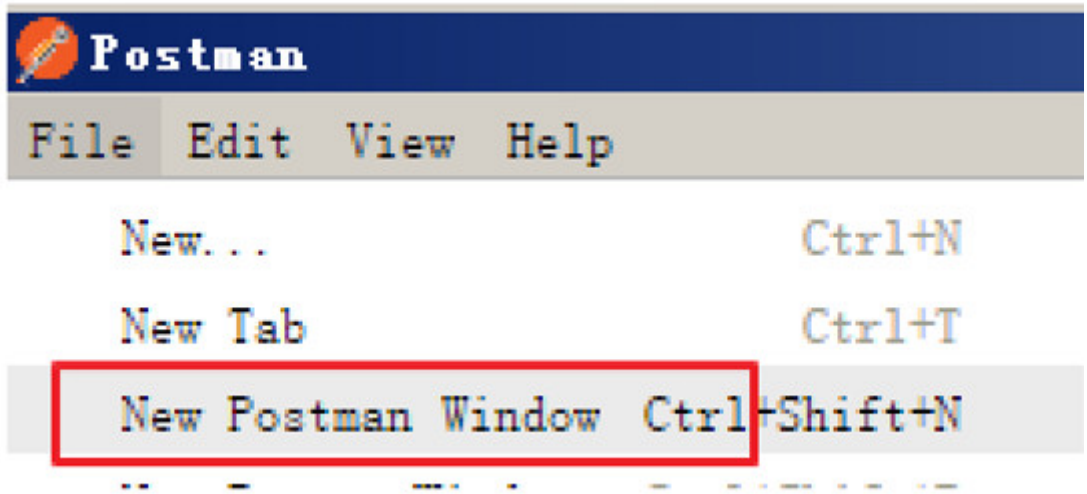
官方地址：https://www.getpostman.com/

安装Postman

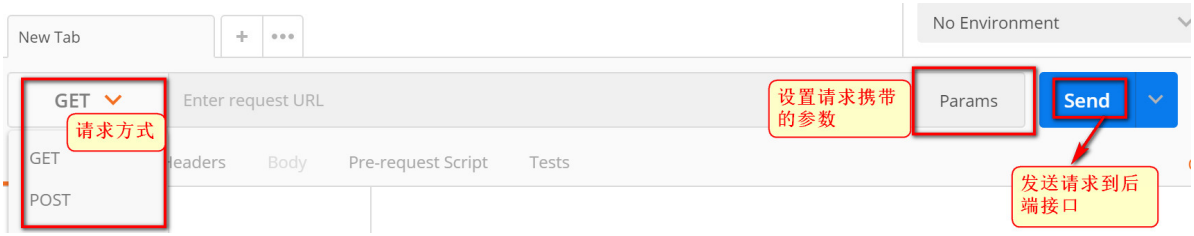
本教程使用，双击打开 Postman-win64-6.0.10-Setup.exe

2.3.2 Postman使用

1. 新建一个Postman窗口



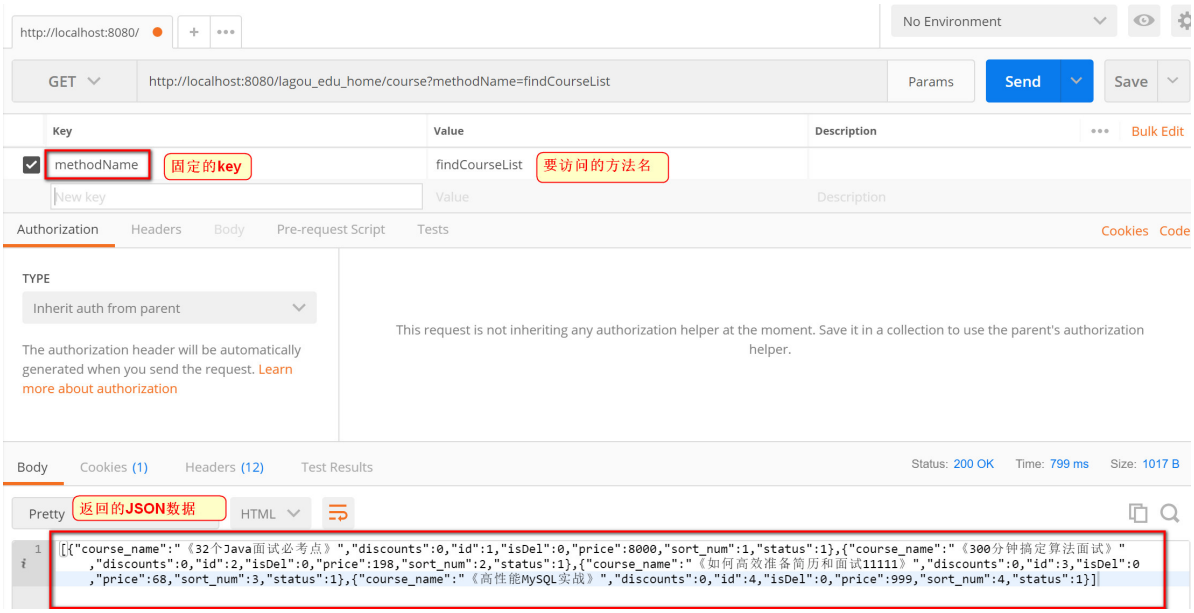
2. 窗口介绍



2.3.3 使用postman测试接口

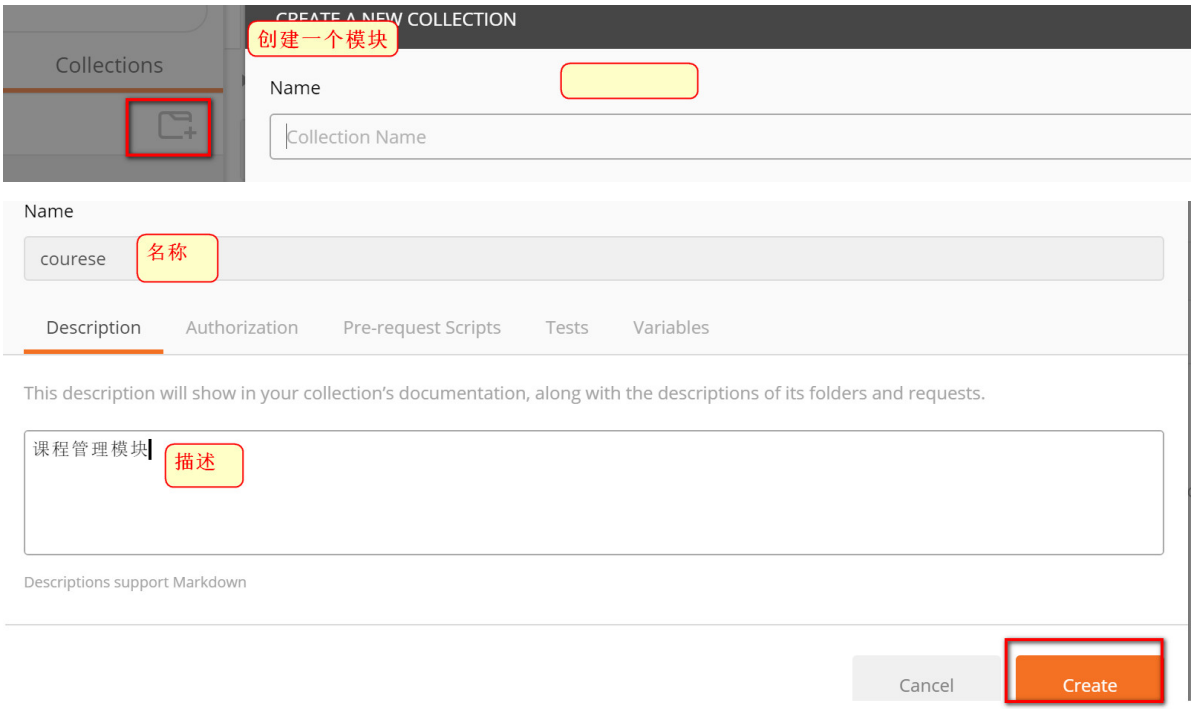
1. 发送请求到指定的

```
http://localhost:8080/lagou_edu_home/course?methodName=findCourseList
```

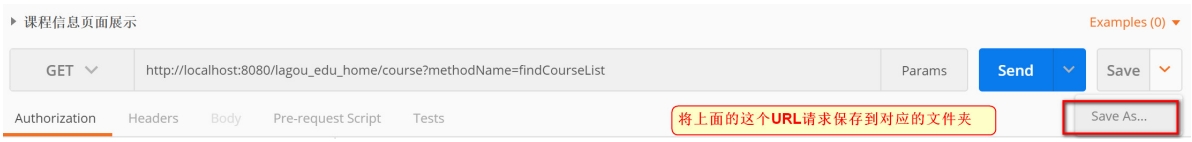


2.3.4 创建模块将请求分类

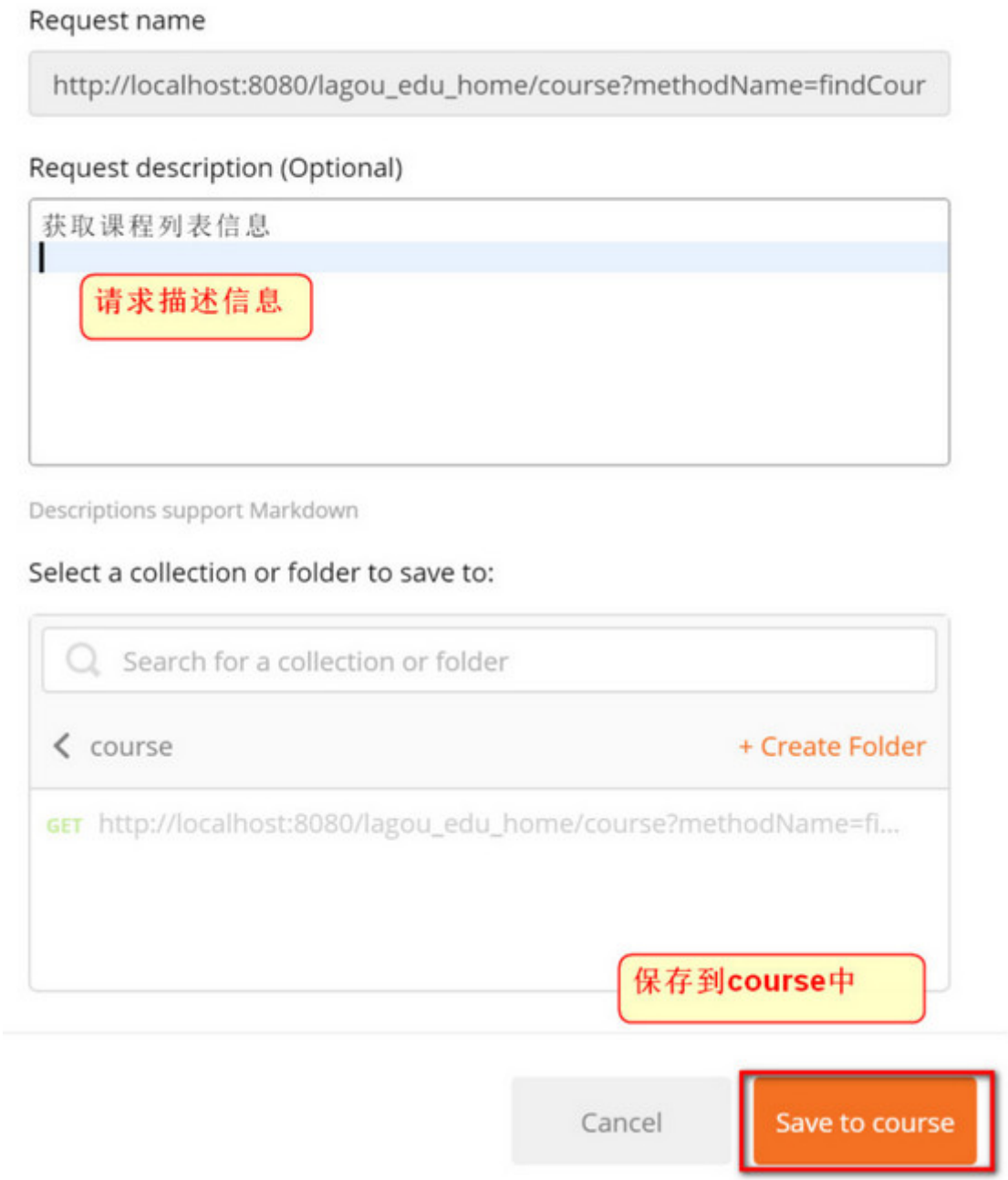
1. 创建课程模块



2. 选择 Save As 将请求保存到对应模块中



3. 描述一下请求的相关信息



3. 功能二: 多条件查询课程信息

3.1 需求分析

1. 根据课程名称和课程状态进行查询

2. 要查询的字段

```
id, course_name,price,  sort_num,   STATUS
```

3. 查询条件

```
is_del
course_name
statuts
```

3.2 根据条件查询课程信息

3.2.2 Dao层编写

1. 因为是多条件查询,所以要注意多个参数情况下,SQL的编写

接口

```
    /**
     * 根据课程名称,课程状态 查询课程信息
     * */
    public List<Course> findByCourseNameAndStatus(String courseName, String
status);
```

实现类

```
    /**
     * 根据课程名称,课程状态 查询课程信息
     * */
    //根据条件查询课程信息
    @Override
    public List<Course> findByCourseNameAndStatus(String courseName, String
status) {

        try {
            //1.创建QueryRunner
            QueryRunner qr = new QueryRunner(DruidUtils.getDataSource());

            //2.编写SQL 当前的查询为多条件不定项查询
            //2.1 创建StringBuffer 对象,将SQL字符串 添加进缓冲区
            StringBuffer sb = new StringBuffer("SELECT
id,course_name,price,sort_num,STATUS FROM course WHERE 1=1 and is_del = ? ");

            //2.2 创建list集合 保存参数
            List<Object> list = new ArrayList<>();
            list.add(0);

            //2.3 判断传入的参数是否为空
            if(courseName != null && courseName != ""){
                sb.append(" AND course_name LIKE ?");
                //like查询 需要拼接 %
                courseName = "%"+courseName+"%";
                //将条件放进list集合
                list.add(courseName);
            }

            if(status != null && status != ""){
                sb.append("AND STATUS = ?");
                //将status 转换为 int
                int i = Integer.parseInt(status);
                list.add(i);
            }
```

```
            //执行查询
            List<Course> courseList = qr.query(sb.toString(), new
BeanListHandler<Course>(Course.class), list.toArray());

            //返回结果
            return courseList;
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        }
    }
```

3.2.3 Service层编写

```
CourseService 接口

    public List<Course> findByCourseNameAndStatus(String courseName, String
status);

CourseServiceImpl 实现类

    @Override
    public List<Course> findByCourseNameAndStatus(String courseName, String
status) {

        return courseDao.findByCourseNameOrStatus(courseName,status);
    }
```

3.2.4 Servlet编写

在CourseServlet中添加 **findByCourseNameOrStatus**方法

```
 //根据条件查询课程信息
    public void findByCourseNameOrStatus(HttpServletRequest request ,
HttpServletResponse response){

        try {
            //1.接收参数
            String courseName = request.getParameter("course_name");
            String status = request.getParameter("status");

            //2.业务处理
            CourseService cs = new CourseServiceImpl();
            List<Course> courseList = cs.findByCourseNameOrStatus(courseName,
status);

            //3.返回结果 响应JSON格式数据
            //使用 SimplePropertyPreFilter,指定要转换为JSON的字段
            SimplePropertyPreFilter filter =
                    new
SimplePropertyPreFilter(Course.class,"id","course_name","price","sort_num","stat
us");
```

```
            String result = JSON.toJSONString(courseList, filter);
            response.getWriter().println(result);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

3.2.5 接口测试

- 请查阅接口文档,使用postman进行接口测试.

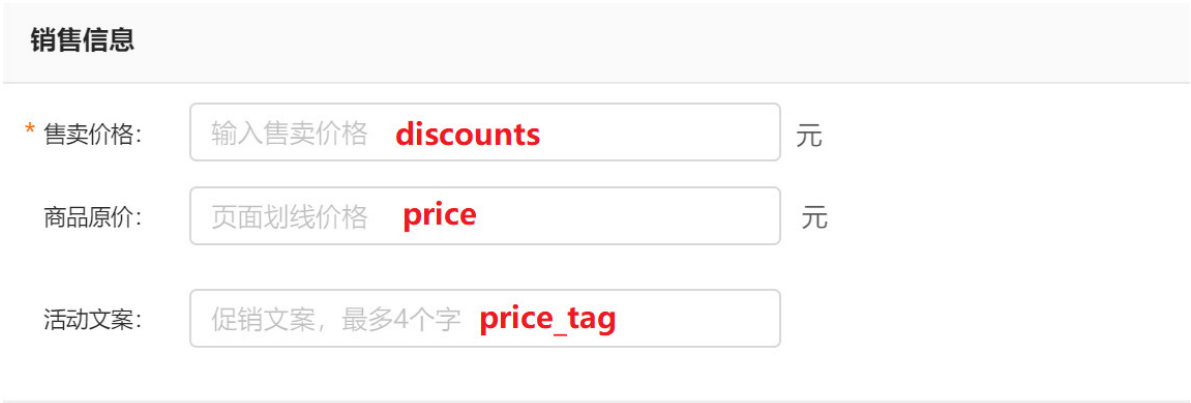
4. 功能三: 新建课程营销信息

4.1 需求分析

1. 选择新建课程,对课程营销信息进行录入

4.1.1 基本信息

4.1.2 销售信息



4.1.3 分享信息

4.1.4 课程详情

4.2 Dao层编写

接口

```
    //保存课程营销信息
    public int saveCourseSalesInfo(Course course);
```

实现类

```
    //保存课程营销信息
    @Override
    public int saveCourseSalesInfo(Course course) {

        try {
            //1.创建QueryRunner
            QueryRunner qr = new QueryRunner(DruidUtils.getDataSource());

            //2.编写SQL
            String sql = "INSERT INTO course(\n" +
```

```
                    "course_name,\n" +
                    "brief,\n" +
                    "teacher_name,\n" +
                    "teacher_info,\n" +
                    "preview_first_field,\n" +
                    "preview_second_field,\n" +
                    "discounts,\n" +
                    "price,\n" +
                    "price_tag,\n" +
                    "share_image_title,\n" +
                    "share_title,\n" +
                    "share_description,\n" +
                    "course_description,\n" +
                    "course_img_url,\n" +
                    "STATUS,\n" +
                    "create_time,\n" +
                    "update_time\n" +
                    ")VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?);";

            //3.准备参数
            Object[] param =
{course.getCourse_name(),course.getBrief(),course.getTeacher_name(),course.getTeacher_info(),

 course.getPreview_first_field(),course.getPreview_second_field(),course.getDiscounts(),course.getPrice(),

 course.getPrice_tag(),course.getShare_image_title(),course.getShare_title(),course.getShare_description(),

 course.getCourse_description(),course.getCourse_img_url(),course.getStatus(),course.getCreate_time(),course.getUpdate_time()};

            //4.执行插入操作
            int row = qr.update(sql, param);

            return row;
        } catch (SQLException e) {
            e.printStackTrace();
            return 0;
        }
    }
```

4.3 Dao层方法测试

```
    //测试保存课程营销信息
    @Test
    public void testSaveCourseSalesInfo(){

        //1.创建course对象
        Course course = new Course();
        course.setCourse_name("爱情36计");
        course.setBrief("学会去找对象");
        course.setTeacher_name("药水哥");
        course.setTeacher_info("人人都是药水哥");
        course.setPreview_first_field("共10讲");
```

```
        course.setPreview_second_field("每周日更新");
        course.setDiscounts(88.88);
        course.setPrice(188.0);
        course.setPrice_tag("最新优惠价");
        course.setShare_image_title("哈哈哈");
        course.setShare_title("嘻嘻嘻");
        course.setShare_description("天天向上");
        course.setCourse_description("爱情36计,就像一场游戏");
        course.setCourse_img_url("https://www.xx.com/xxx.jpg");
        course.setStatus(1); //1 上架 ,0 下架
        String formart = DateUtils.getDateFormart();
        course.setCreate_time(formart);
        course.setUpdate_time(formart);

        int i = courseDao.saveCourseSalesInfo(course);
        System.out.println(i);
    }
```

4.4 Service层编写

1.编写枚举类,设置响应状态码

```
public enum StatusCode {

    SUCCESS(0,"success"),
    FAIL(1,"fail");

    //定义属性
    private int code;
    private String message;

    //定义构造
    StatusCode(int code, String message) {
        this.code = code;
        this.message = message;
    }

    //get/set
    public int getCode() {
        return code;
    }

    public void setCode(int code) {
        this.code = code;
    }

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }
```

```
    //重写toString,将枚举对象转化为JSON
    @Override
    public String toString() {
        JSONObject object = new JSONObject();
        object.put("status",code);
        object.put("msg",message);
        return object.toString();
    }
}
```

2. 编写Service

接口

```
    public String saveCourseSalesInfo(Course course);
```

实现类

```
    @Override
    public String saveCourseSalesInfo(Course course) {
        //1.补全课程信息
        String dateFormart = DateUtils.getDateFormart();
        course.setCreate_time(dateFormart);
        course.setUpdate_time(dateFormart);
        course.setStatus(0);

        //2.调用Dao进行插入
        int i = courseDao.saveCourseSalesInfo(course);
        if(i > 0){
            //保存成功
            String result = StatusCode.SUCCESS.toString();
            return result;

        }else{
            //保存失败
            String result = StatusCode.FAIL.toString();
            return result;
        }
    }
```

4.5 文件上传

4.5.1 图片上传分析

在添加课程营销信息的表单中,有一个图片上传项

4.5.2 文件上传介绍

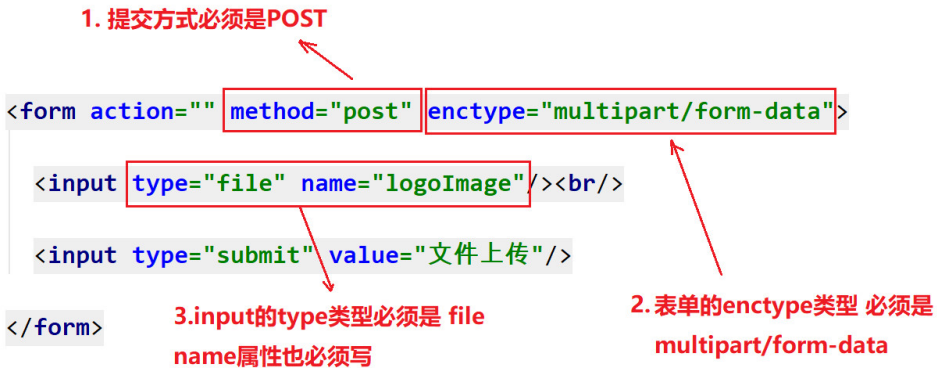
文件上传的实质：文件的拷贝

- 文件上传：从本地将文件拷贝到服务器磁盘上
  - 客户端：需要编写文件上传表单
  - 服务端：需要编写代码接受上传的 文件

4.5.3 客户端编码

1. 文件上传三要素:

- 1.表单提交方式: **post** (get方式提交有大小限制,post没有)
- 2.表单的enctype属性:必须设置为 **multipart/form-data.**
  - enctype就是encodetype就是编码类型的意思.
  - multipart/form-data是多部件文件上传，指表单数据有多部分构成，既有文本数据，又有文件等二进制数据的意思。
- 3.表单必须有文件上传项: **file** ,必须要有name属性和值



> 注意：默认情况下，表单的enctype的值是application/x-www-form-urlencoded，不能用于文件上传，只有使用了multipart/form-data，才能完整的传递文件数据

2. 代码示例

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <%--
        表单提交必须是POST ,
        表单的enctype属性:必须设置为    multipart/form-data.
        input的type类型必须指定为: file, 一定要有name属性
        --%>
    <form action="${pageContext.request.contextPath}/upload" method="post"
enctype="multipart/form-data">
        <input type="file" name="upload">
        <br>
        <input type="text" name="name">
        <input type="text" name="password">
        <input type="submit" value="文件上传">
    </form>
```

```
</body>
</html>
```

4.5.4 服务端编码

服务端要接收文件上传的表单数据

1. 上传文件, 抓包分析

使用360浏览器进行抓包,谷歌浏览器不方便查看

2. 服务端获上传的文件

1. 通过request获取请求体的内容
2. 解析请求体 多部件上传的特点是,每个input都是一个表单项.

   根据分隔符将请求中所有的内容,切割成数组,数组中的每一个元素 都是一个表单项
3. 遍历数组,分清楚那个是普通的表单项, 哪个是 文件上传项

   如何区分? 判断是否有 filename
4. 获取到普通表单项中的内容,通过属性name获取
5. 获取文件上传项内容

   文件名: filname = aaa.txt

   文件内容:
6. 使用IO将文件内容,保存到服务器中

4.5.5 FileUpload工具类

1. 导入依赖

FileUpload包可以很容易地将文件上传到你的Web应用程序.

IOUtils封装了Java中io的常见操作，使用十分方便 ,需要下载 commons-io-1.4.jar 包

```
<dependency>
    <groupId>commons-io</groupId>
    <artifactId>commons-io</artifactId>
    <version>1.4</version>
</dependency>

<dependency>
    <groupId>commons-fileupload</groupId>
    <artifactId>commons-fileupload</artifactId>
    <version>1.2.1</version>
</dependency>
```

2. FileUpload 核心类介绍

类名	介绍
DiskFileItemFactory	磁盘文件项工厂, 读取文件时相关的配置,比如: 缓存的大小 , 临时目录的位置
ServletFileUplaod	文件上传的一个核心类
FileItem	代表每一个表单项

3. 文件上传的API的详解

- ServletFileUpload

方法	说明
isMultipartContent(request);	判断是否是一个文件上传的表单
parseRequest(request);	解析request获得表单项的集合
setHeaderEncoding("UTF-8");	设置上传的文件名的编码方式

- FileItem

方法	说明
isFormField()	判断是否是普通表单项
getFieldName()	获得表单的name属性值
item.getString()	获得表单的value值
getName()	获得上传文件的名称
getInputStream()	获得上传文件
delete()	删除临时文件

4. 文件上传后台代码编写

FileUpload使用步骤:

1、创建磁盘文件项工厂

2、创建文件上传的核心类

3、解析request---获得文件项集合

4、遍历文件项集合

5、判断普通表单项/文件上传项

```
@WebServlet("/upload")
public class FileUploadServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {

        try {
            //1.创建磁盘文件项工厂
            DiskFileItemFactory factory = new DiskFileItemFactory();

            //2.创建文件上传核心类
            ServletFileUpload upload = new ServletFileUpload(factory);
            //2.1 设置上传文件名的编码
            upload.setHeaderEncoding("utf-8");
            //2.2 判断表单是否是文件上传表单
            boolean multipartContent = upload.isMultipartContent(req);
            //2.3 是文件上传表单
            if(multipartContent){

                //3. 解析request ,获取文件项集合
                List<FileItem> list = upload.parseRequest(req);

                if(list != null){
                    //4.遍历获取表单项
                    for (FileItem item : list) {
                        //5. 判断是不是一个普通表单项
                        boolean formField = item.isFormField();
                        if(formField){
                            //普通表单项, 当 enctype="multipart/form-data"时,
request的getParameter()方法 无法获取参数
                            String fieldName = item.getFieldName();
                            String value = item.getString("utf-8");//设置编码
                            System.out.println(fieldName + "=" + value);
                        }else{

                            //文件上传项
                            //文件名
                            String fileName = item.getName();

                            //避免图片名重复 拼接UUID
                            String newFileName = UUIDUtils.getUUID()+"_"+
fileName;

                            //获取输入流
```

```
                            InputStream in = item.getInputStream();

                            //创建输出流 输出到H盘
                            FileOutputStream fos = new
FileOutputStream("H:/upload/" +newFileName);

                            //使用工具类IOUtils,copy文件
                            IOUtils.copy(in,fos);

                            //关闭流
                            fos.close();
                            in.close();

                        }

                    }
                }
            }
        } catch (FileUploadException e) {
            e.printStackTrace();
        }

    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        doGet(req, resp);
    }

}
```

4.5.6 将图片上传到tomcat服务器

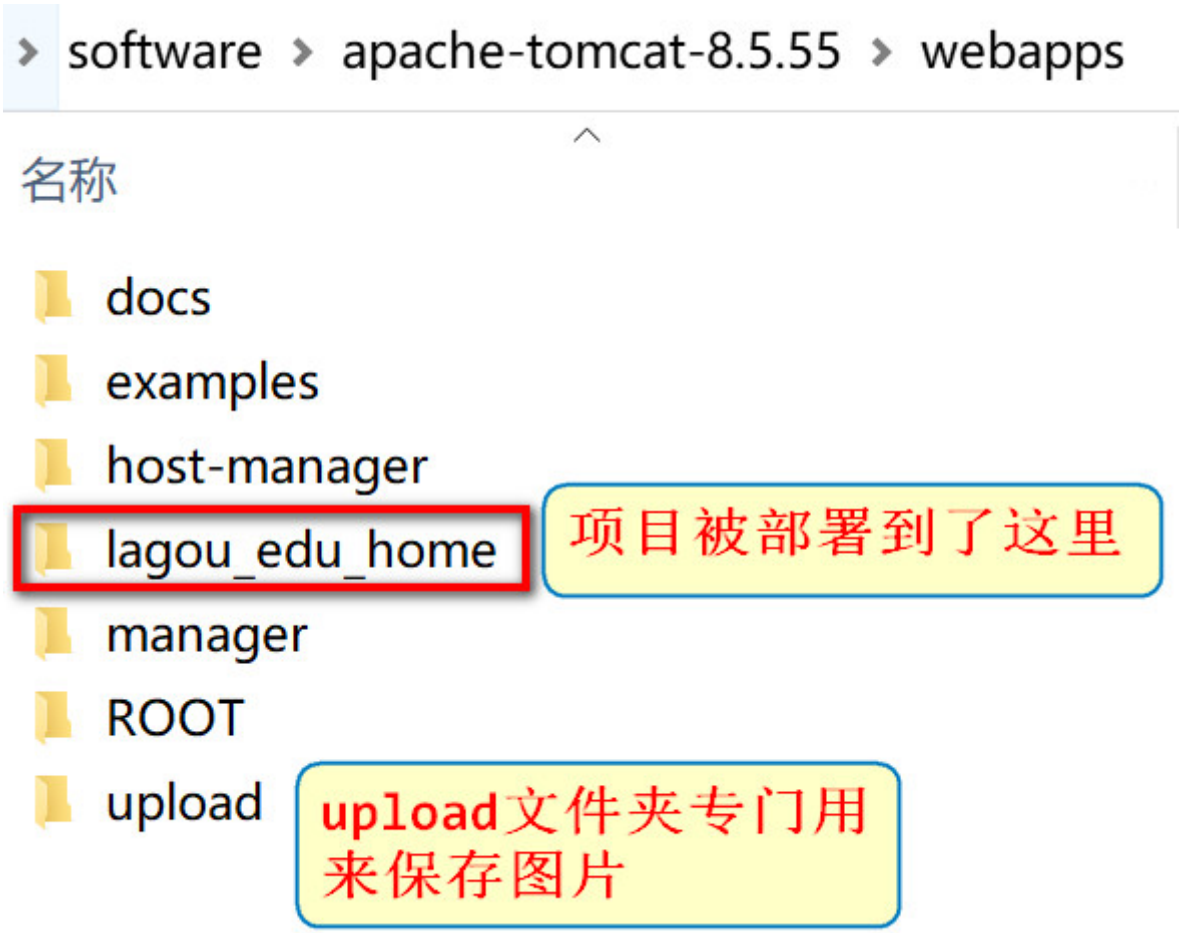
1. 将项目部署到webapps

将部署方式改变为 war模式,把项目部署在tomcat的webapps下

- idea中部署项目两种方式
  - war模式：将项目以war包的形式上传真实到服务器的webapps目录中；
  - war exploded模式：仅仅是目录的映射，就相当于tomcat在项目源文件夹中启动一样；

2.在webapps中创建upload目录

upload目录专门用来保存上传过来的图片



3.修改代码,将图片上传到服务器

- 修改图片的输出路径
    1. 获取到项目的运行目录信息
    2. 截取到webapps的 目录路径
    3. 拼接输出路径,将图片保存到upload

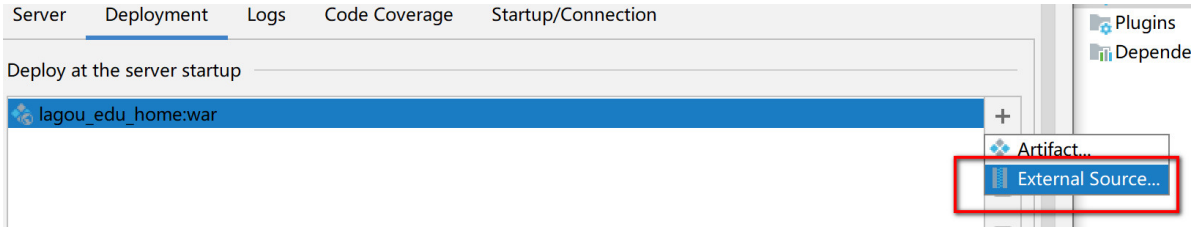
```
@Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {

        try {
            //1.创建磁盘文件项工厂
            DiskFileItemFactory factory = new DiskFileItemFactory();

            //2.创建文件上传核心类
            ServletFileUpload upload = new ServletFileUpload(factory);
            //2.1 设置上传文件名的编码
            upload.setHeaderEncoding("utf-8");
            //2.2 判断表单是否是文件上传表单
            boolean multipartContent = upload.isMultipartContent(req);
            //2.3 是文件上传表单
            if(multipartContent){

                //3. 解析request ,获取文件项集合
                List<FileItem> list = upload.parseRequest(req);
```

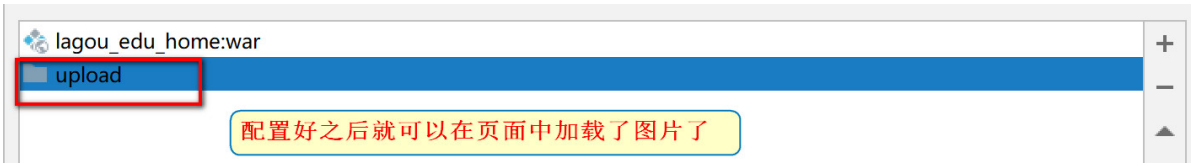
```
                if(list != null){
                    //4.遍历获取表单项
                    for (FileItem item : list) {
                        //5. 判断是不是一个普通表单项
                        boolean formField = item.isFormField();
                        if(formField){
                            //普通表单项, 当 enctype="multipart/form-data"时,
request的getParameter()方法 无法获取参数
                            String fieldName = item.getFieldName();
                            String value = item.getString("utf-8");//设置编码
                            System.out.println(fieldName + "=" + value);
                        }else{

                            //文件上传项
                            //文件名
                            String fileName = item.getName();
                            //避免图片名重复 拼接UUID
                            String newFileName = UUIDUtils.getUUID()+"_"+
fileName;

                            //获取上传文件的内容
                            InputStream in = item.getInputStream();

                            String path =
this.getServletContext().getRealPath("/");

                            //获取到 webapps路径
                            String webappsPath = path.substring(0,
path.indexOf("lagou_edu_home"));
                            OutputStream out = new
FileOutputStream(webappsPath+"/upload/"+newFileName);
                            //拷贝文件到服务器
                            IOUtils.copy(in,out);

                            out.close();
                            in.close();
                        }

                    }
                }
            }
        } catch (FileUploadException e) {
            e.printStackTrace();
        }

    }
```

4. 页面加载图片

将tomcat作为图片服务器使用时，存储上传的图片后,如果想要图片可以访问,需要在idea中进行配置:

1. 选择external source ---> 找到webapps目录下的的upload文件夹





2. 上传一张图片到服务器
3. 在项目内部页面加载图片

```
<img src="/upload/abbd99891af442a8a9cb65848744452e_qiyu.jpg">
```

4. 也可以通过HTTP方式访问

```
http://localhost:8080/upload/abbd99891af442a8a9cb65848744452e_qiyu.jpg
```

4.6 BeanUtils工具类

1. 介绍

BeanUtils 是 Apache commons组件的成员之一，主要用于简化JavaBean封装数据的操作。**可以将一个表单提交的所有数据封装到JavaBean中。**

2. 导入依赖

```
<dependency>
    <groupId>commons-beanutils</groupId>
    <artifactId>commons-beanutils</artifactId>
    <version>1.8.3</version>
</dependency>
```

3. BeanUtils 对象常用方法

方法	描述
populate(Object bean, Map properties)	将Map数据封装到指定Javabean中，一般用于将表单的所有数据封装到javabean
setProperty(Object obj,String name,Object value)	设置属性值
getProperty(Object obj,String name)	获得属性值

4. BeanUtils 使用测试

```
public class TestBeanUtils {
```

```
    @Test
    public void test01() throws InvocationTargetException,
IllegalAccessException, NoSuchMethodException {

        //1.创建course对象
        Course course = new Course();

        //2.创建Map
        Map<String,Object> map = new HashMap<>();

        //3.向map集合中添加数据, key要与course的属性名保持一致,value的数据类型与course的
属性的类型保持一致
        map.put("id",1);
        map.put("course_name","大数据");
        map.put("brief","课程包含所有大数据流行的技术");
        map.put("teacher_name","周星星");
        map.put("teacher_info","非著名演员");

        //将map中的数据封装到 course中
        BeanUtils.populate(course,map);

        System.out.println(course.getId()+" " + course.getCourse_name() +" "
+course.getBrief()
        +"   "+course.getTeacher_name()+" " +course.getTeacher_info());

        //设置属性 获取属性
        BeanUtils.setProperty(course,"price",100.0);

        String price = BeanUtils.getProperty(course, "price");

        System.out.println(price);
    }
}
```

4.7 Servlet编写

4.7.1 CourseSalesInfoServlet

创建CourseSalesInfoServlet类,继承HttpServlet , 完成保存课程营销信息操作.

因为上传的信息包含文件信息,无法直接通过request直接获取参数,所以不能继承BaseServlet

```
@WebServlet("/courseSalesInfo")
public class CourseSalesInfoServlet  extends HttpServlet {

    /**
     * 保存课程营销信息
     *       收集表单数据,封装到course对象中,将图片上传到tomcat服务器中
     * */
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {

        try {
            //1.创建Course对象
```

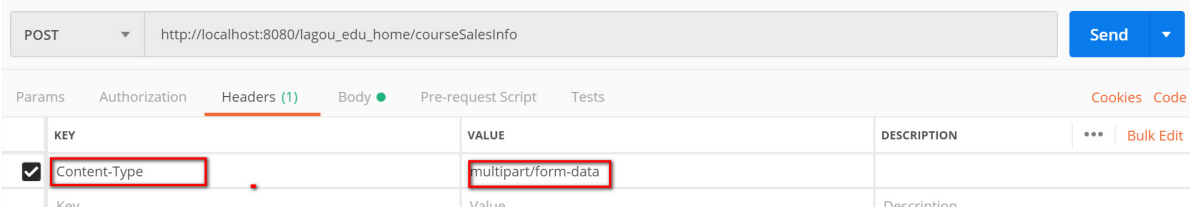
```
            Course course = new Course();

            //2.创建Map集合,用来收集数据
            Map<String,Object> map = new HashMap<>();

            //3.创建磁盘工厂对象
            DiskFileItemFactory factory = new DiskFileItemFactory();

            //4.文件上传核心对象
            ServletFileUpload fileUpload = new ServletFileUpload(factory);

            //5.解析request对象,获取表单项集合
            List<FileItem> list = fileUpload.parseRequest(req);

            //6.遍历集合 判断哪些是普通的表单项,那些是文件表单项
            for (FileItem item : list) {

                boolean formField = item.isFormField();
                if(formField){
                    //是普通表单项,获取表单项中的数据,保存到map
                    String fieldName = item.getFieldName();
                    String value = item.getString("UTF-8");
                    System.out.println(fieldName +" " + value);
                    //使用map收集数据
                    map.put(fieldName,value);
                }else{
                    //文件上传项
                    //获取文件名
                    String fileName = item.getName();
                    String newFileName = UUIDUtils.getUUID()+"_"+fileName;

                    //获取输入流
                    InputStream in = item.getInputStream();

                    //获取webapps的目录路径
                    String realPath = this.getServletContext().getRealPath("/");
                    String wabappsPath = realPath.substring(0,
realPath.indexOf("lagou_edu_home"));

                    //创建输出流
                    OutputStream out = new
FileOutputStream(wabappsPath+"/upload/" + newFileName);

                    IOUtils.copy(in,out);
                    out.close();
                    in.close();

                    //将图片路径进行保存
                    map.put("course_img_url", Constants.LOCAL_URL+"/upload/" +
newFileName);
                }
            }

            //使用BeanUtils 将map中的数据封装到course对象
            BeanUtils.populate(course,map);

            String dateFormart = DateUtils.getDateFormart();
            CourseService cs = new CourseServiceImpl();
```

```
            if(map.get("id") != null){
                //修改操作
                //补全信息
                course.setUpdate_time(dateFormart);//修改时间
                String result = cs.updateCourseSalesInfo(course);
                //响应结果
                resp.getWriter().print(result);

            }else{
                //新建操作
                //补全信息
                course.setCreate_time(dateFormart);//创建时间
                course.setUpdate_time(dateFormart);//修改时间
                course.setStatus(1); //上架
                String result = cs.saveCourseSalesInfo(course);
                //响应结果
                resp.getWriter().print(result);
            }

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

    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        doGet(req, resp);
    }
}
```

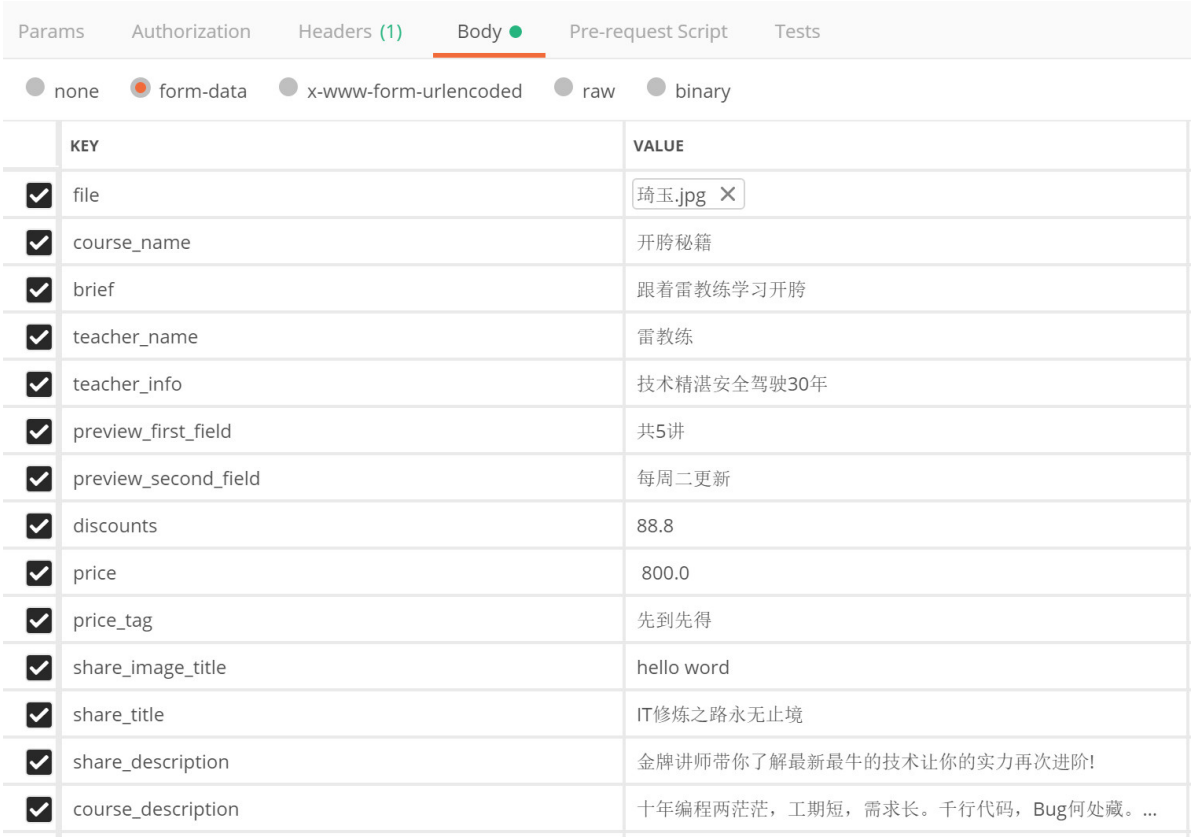
4.7.2 接口测试

postman测试上传文件

1. 接口地址填写正确
2. 将请求方式设置为POST
3. 需要上传文件, 设置Headers: **"key":"Content-Type", "value":"multipart/form-data"**



4. Body选择form-data
5. key 右侧下拉选择file；value 点击Select Files选择文件 , 按照接口文档,补全测试参数



4.7.3 保存图片URL优化

1.创建常量类

```
public final class Constants {

    //本地访问地址
    public static final String LOCAL_URL = "http://localhost:8080";
}
```

2.拼接图片URL

```
//将图片路径进行保存
map.put("course_img_url", Constants.LOCAL_URL+"/upload/" + newFileName);
```

5. 功能四: 修改课程营销信息

5.1 需求分析

营销信息其实就是课程相关的信息, 操作的依然是 **course** 表. 我们通过点击营销信息按钮,进入到对应的课程营销信息页面,对原有信息进行修改.

5.2 Dao层编写

1. 通过上面的分析,首先要编写 根据课程ID查询课程信息,进行回显

接口

```
    //根据课程ID 查询课程信息
    public Course findCourseById(int id);
```

实现类

```
    //根据课程ID 查询课程营销信息
    @Override
    public Course findCourseById(int id) {
        try {
            QueryRunner qr = new QueryRunner(DruidUtils.getDataSource());

            String sql = "SELECT \n" +
                    "id,\n" +
                    "course_name,\n" +
                    "brief,\n" +
                    "teacher_name,\n" +
                    "teacher_info,\n" +
                    "preview_first_field,\n" +
                    "preview_second_field,\n" +
                    "discounts,\n" +
                    "price,\n" +
                    "price_tag,\n" +
                    "course_img_url,\n" +
                    "share_image_title,\n" +
                    "share_title,\n" +
                    "share_description,\n" +
                    "course_description,\n" +
                    "STATUS\n" +
```

```
                    "FROM course WHERE id = ?;";

            Course course = qr.query(sql, new BeanHandler<Course>(Course.class),
id);
            return course;

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

    }
```

```
-- 根据ID查询课程信息SQL
SELECT
id,
course_name,
brief,
teacher_name,
teacher_info,
preview_first_field,
preview_second_field,
discounts,
price,
price_tag,
course_img_url,
share_image_title,
share_title,
share_description,
course_description,
STATUS
FROM course WHERE id = ?;
```

2. 编写修改课程营销信息的方法,将修改写入数据库

接口

```
    //修改课程营销信息
    public int updateCourseSalesInfo(Course course);
```

实现类

```
    //修改课程营销信息
    @Override
    public int updateCourseSalesInfo(Course course) {

        try {
            QueryRunner qr = new QueryRunner(DruidUtils.getDataSource());

            String sql = "UPDATE course SET \n" +
                    "course_name = ?,\n" +
                    "brief = ?,\n" +
                    "teacher_name = ?,\n" +
                    "teacher_info = ?,\n" +
                    "preview_first_field = ?,\n" +
                    "preview_second_field = ?,\n" +
                    "discounts = ?,\n" +
                    "price = ?,\n" +
```

```
                    "price_tag = ?,\n" +
                    "share_image_title = ?,\n" +
                    "share_title = ?,\n" +
                    "share_description = ?,\n" +
                    "course_description = ?,\n" +
                    "course_img_url = ?,\n" +
                    "update_time = ?\n" +
                    "WHERE id = ?";

            Object[] param =
{course.getCourse_name(),course.getBrief(),course.getTeacher_name(),course.getTeacher_info(),

 course.getPreview_first_field(),course.getPreview_second_field(),course.getDiscounts(),course.getPrice(),course.getPrice_tag(),

 course.getShare_image_title(),course.getShare_title(),course.getShare_description(),course.getCourse_description(),
            course.getCourse_img_url(),course.getUpdate_time(),course.getId()};

            int row  = qr.update(sql, param);
            return row;

        } catch (SQLException e) {
            e.printStackTrace();
            return 0;
        }

    }
```

修改课程

```
UPDATE course SET
course_name = ?,
brief = ?,
teacher_name = ?,
teacher_info = ?,
preview_first_field = ?,
preview_second_field = ?,
discounts = ?,
price = ?,
price_tag = ?,
share_image_title = ?,
share_title = ?,
share_description = ?,
course_description = ?,
course_img_url = ?,
update_time = ?
WHERE id = ?
```

3. 测试

5.3 Service层编写

```
接口
    public Course findCourseById(int id);

实现类
    @Override
    public Course findCourseById(int id) {
        return courseDao.findCourseById(id);
    }
```

```
接口
    public String updateCourseSalesInfo(Course course);

实现类
     @Override
    public String updateCourseSalesInfo(Course course)  {

        //调用dao
        int i = courseDao.updateCourseSalesInfo(course);

        //根据插入是否成功,封装对应信息

        if(i > 0){
            //保存成功
            String result = StatusCode.SUCCESS.toString();
            return result;
        }else{
            //保存失败
            String result = StatusCode.FAIL.toString();
            return result;
        }

    }
```

5.4 Servlet编写

5.4.1 根据ID查询课程信息

5.4.1.1 CourseServlet

在CourseServlet中, 添加根据ID查询课程信息的功能

```
    /**
     * 根据课程ID查询课程营销信息
     * */
    public void findCourseById(HttpServletRequest request , HttpServletResponse
response){

        try {
            //1.接收参数
            String id = request.getParameter("id");

            //2.业务处理
            CourseService cs = new CourseServiceImpl();
            Course course = cs.findCourseById(Integer.parseInt(id));
```

```
            //3.返回结果 响应JSON格式数据
            //使用 SimplePropertyPreFilter,指定要转换为JSON的字段
            SimplePropertyPreFilter filter = new
SimplePropertyPreFilter(Course.class,"id","course_name","brief","teacher_name",

 "teacher_info","preview_first_field","preview_second_field","discounts","price"
,"price_tag","share_image_title","share_title","share_description","course_descr
iption");

            String result  = JSON.toJSONString(course, filter);
            response.getWriter().println(result);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

5.4.1.2 接口测试

详见接口文档

5.4.2 修改CourseSalesInfoServlet

5.4.2.1 需求分析

保存营销信息和修改营销信息,访问的是同一个接口,所以在**CourseSalesInfoServlet**中,我们需要进行一下判断

- 携带id 就是修改操作
- 未携带id就是新增操作

5.4.2.2 代码修改

```
@WebServlet("/courseSalesInfo")
public class CourseSalesInfoServlet extends HttpServlet {


    /**
     *  保存营销信息
     *      收集表单的数据 封装一个Course实体 将上传图片存到服务器磁盘上
     * */
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {

        try {
            //1.获取参数,调用FIleUploadUtils,进行文件上传和参数的封装
            Map<String, Object> map = FileUploadUtil.upload(req);

            //2.使用BeanUtils 将map中的数据封装到 Course对象中
            Course course = new Course();
            BeanUtils.populate(course,map);

            //3.业务处理
            if(map.get("id") != null){
```

```
                //补全信息 修改时间
                course.setUpdate_time(DateUtils.getDateFormart());

                CourseService cs = new CourseServiceImpl();
                Map<String, String> message = cs.updateSalesInfo(course);

                //4.响应JSON数据
                String result = JSON.toJSONString(message);
                resp.getWriter().println(result);

            }else{

                //补全信息
                course.setCreate_time(DateUtils.getDateFormart());//创建时间
                course.setUpdate_time(DateUtils.getDateFormart());//修改时间
                course.setStatus(0);//状态

                //8.业务处理
                CourseService cs = new CourseServiceImpl();
                Map<String, String> message = cs.saveSalesInfo(course);

                //9.响应JSON数据
                String result = JSON.toJSONString(message);
                resp.getWriter().println(result);
            }

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

    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        doGet(req, resp);
    }

}
```

5.4.2.3 接口测试

根据接口文档,进行测试

6. 功能五: 修改课程状态

6.1 需求分析

1. 数据库中课程状态码为0或者1 ,课程状态，0-草稿(下架)，1-上架

```
status                  int(2) NULL                     课程状态，0-草稿，1-上架
```

2. 页面分析

6.2 DAO层编写

接口

```
    //修改课程状态
    int updateCourseStatus(Course course);
```

实现类

```
    //修改课程状态
    @Override
    public int updateCourseStatus(Course course) {

        try {
            QueryRunner qr = new QueryRunner(DruidUtils.getDataSource());

            String sql = "UPDATE course SET STATUS = ? ,update_time = ? WHERE id
= ?";

            Object[] param =
{course.getStatus(),course.getUpdate_time(),course.getId()};

            int row = qr.update(sql, param);

            return row;
        } catch (SQLException e) {
            e.printStackTrace();
            return 0;
        }
    }
```

6.3 Service层编写

接口

```
    public Map<String,Integer> updateCourseStatus(Course course);
```

实现类

```
    @Override
    public Map<String, Integer> updateCourseStatus(Course course) {

        //调用dao
        int row = courseDao.updateCourseStatus(course);

        Map<String ,Integer> map = new HashMap<>();

        if(row > 0){

            if(course.getStatus() == 0){
                map.put("status",0);
```

```
            }else{
                map.put("status",1);
            }
        }

        return map;
    }
```

6.4 Servlet编写

在CourseServlet中, 添加**updateCourseStatus**方法

```
    //修改课程状态
    public void updateCourseStatus(HttpServletRequest
request,HttpServletResponse response){

        try {
            //1.获取参数
            String id = request.getParameter("id");

            //2.业务处理
            CourseService cs = new CourseServiceImpl();

            //3.根据课程id 查询课程信息
            Course course = cs.findCourseById(Integer.parseInt(id));

            //4.判断课程信息状态,进行取反设置
            int status = course.getStatus();
            if(status == 0){
                //如果是0 设置为1
                course.setStatus(1);
            }else{
                course.setStatus(0);
            }

            //5.设置更新时间
            course.setUpdate_time(DateUtils.getDateFormart());

            //6.修改状态
            Map<String, Integer> map = cs.updateCourseStatus(course);

            //7.响应结果
            String result = JSON.toJSONString(map);

            response.getWriter().print(result);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

6.5 接口测试

查看接口文档,进行测试