任务三：拉勾教育后台管理系统（SSM）

任务三主要课程内容：

权限模块

一 权限概念介绍

权限：**权利**(能做的)和**限制**(不能做的)，在权限范围内做好自己的事情，不该看的不看，不该做的不做

认证： 验证用户名密码是否正确的过程

授权： 对用户所能访问的资源进行控制（动态显示菜单、url级别的权限控制）

为什么要实现权限系统

首先系统需要进行登陆才能访问

其次不同登陆用户要有不同的权利，而且要有不同的菜单（例如财务经理针对系统中财务相关模块进行操作，人事经理针对系统中人事模块进行操作）

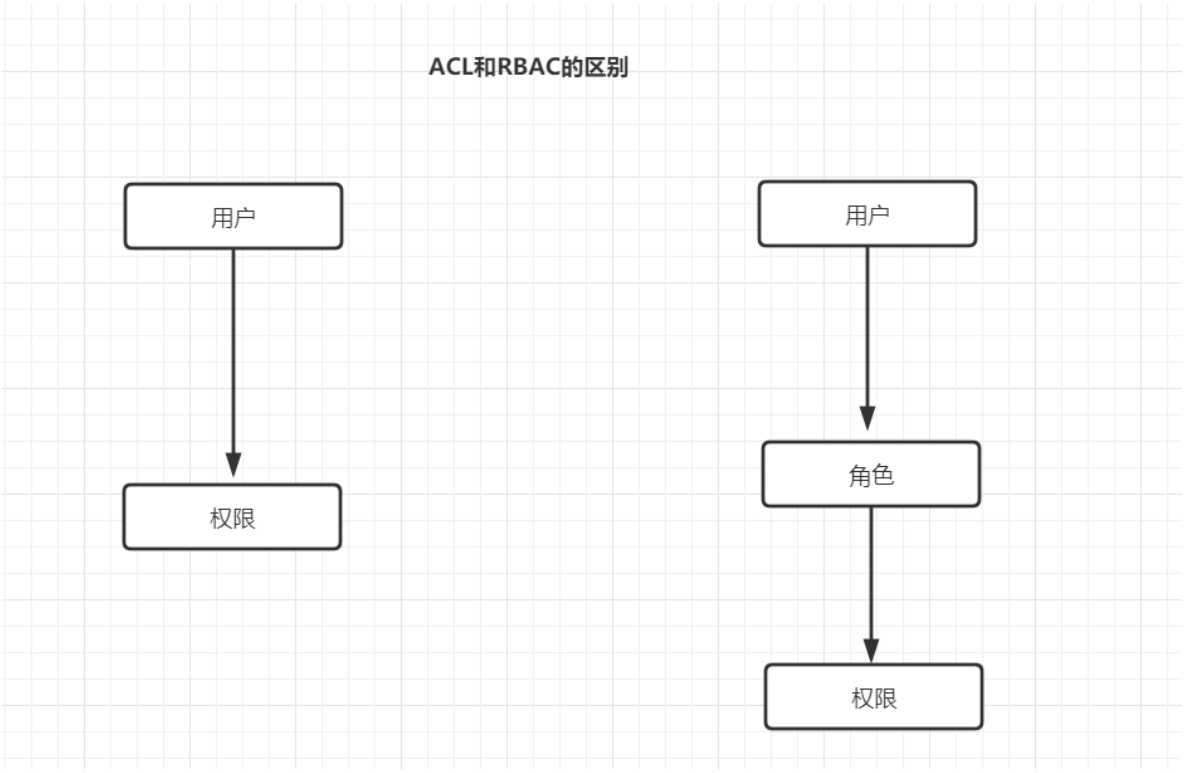
权限控制基本原理

1.ACL(Access Control Lists,缩写ACL)

ACL是最早也是最基本的一种访问控制机制，它的原理非常简单：每一项资源，都配有一个列表，这个列表记录的就是哪些用户可以对这项资源执行CRUD中的那些操作。当系统试图访问这项资源时，会首先检查这个列表中是否有关于当前用户的访问权限，从而确定当前用户可否执行相应的操作。总得来说，ACL是一种面向资源的访问控制模型，它的机制是围绕"资源"展开的。

2.基于角色的访问控制RBAC(Role-Based Access Control)

RBAC是把用户按角色进行归类，通过用户的角色来确定用户能否针对某项资源进行某项操作。RBAC相对于ACL最大的优势就是它简化了用户与权限的管理，通过对用户进行分类，使得角色与权限关联起来，而用户与权限变成了间接关联。RBAC模型使得访问控制，特别是对用户的授权管理变得非常简单和易于维护，因此有广泛的应用



规则一：每个登陆的用户，可以有多个角色；

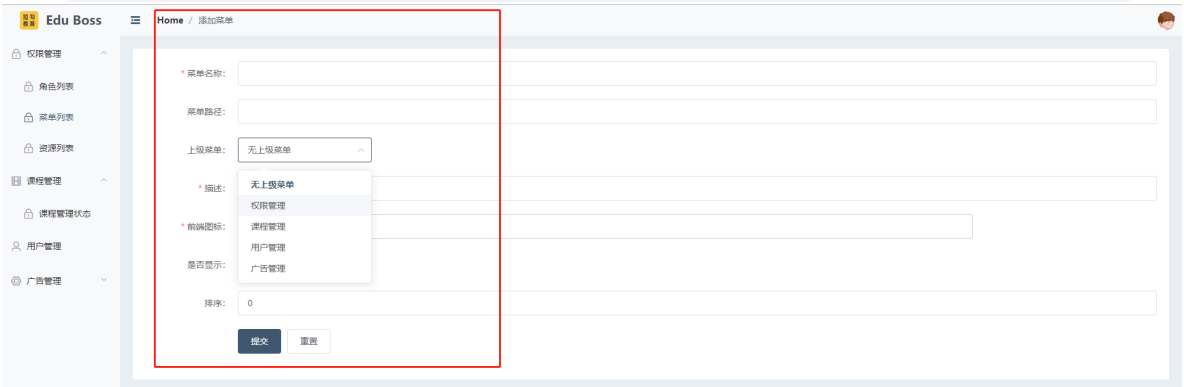
规则二：每个角色又可以拥有多个权限（包含菜单和资源）；

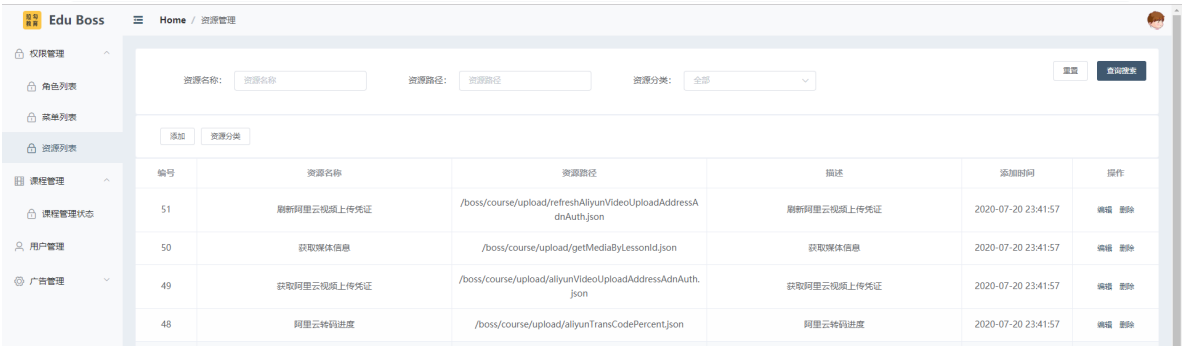
二 权限模块功能分析

权限模块主要细分为角色模块、菜单模块、资源模块，将针对细分的三个模块进行具体功能实现，同时会完成用户登陆、用户关联角色及动态菜单显示

2.1 权限模块管理

- 实现以下功能:
  - 角色列表&条件查询（角色模块）
  - 分配菜单（角色模块）
  - 删除角色（角色模块）
  - 菜单列表查询（菜单模块）
  - 查询菜单信息回显（菜单模块）
  - 资源分页&多条件查询（资源模块）
  - 用户登陆（用户模块）
  - 动态菜单展示（权限模块）
  - 用户关联角色（用户模块）

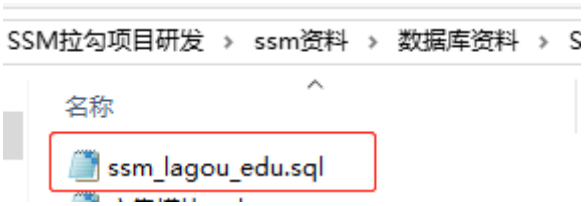




三 权限管理模块表设计

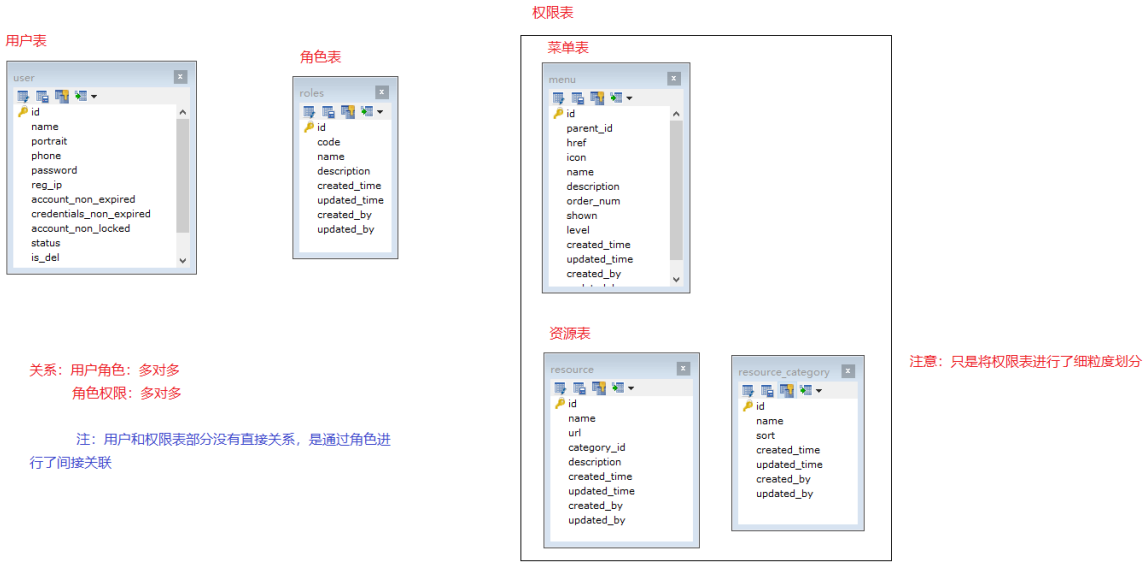
3.1 创建数据库及表

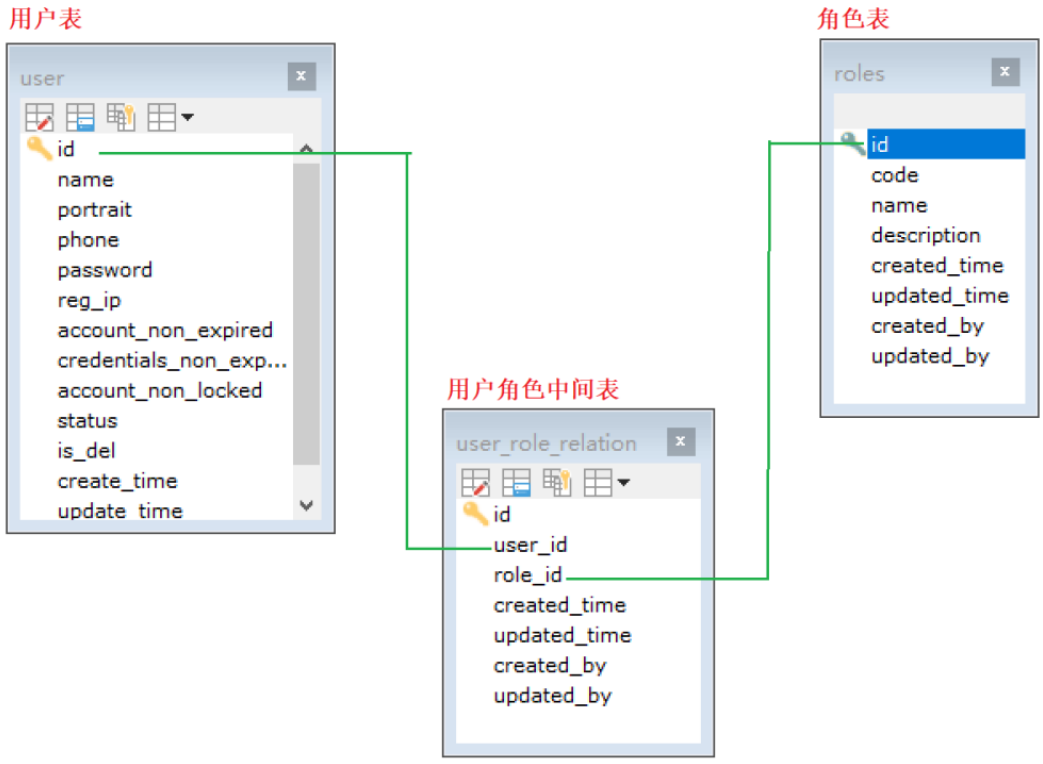
在资料中找到 ssm_lagou_edu.sql,使用SQLYog 执行SQL脚本 ,导入表结构及表信息

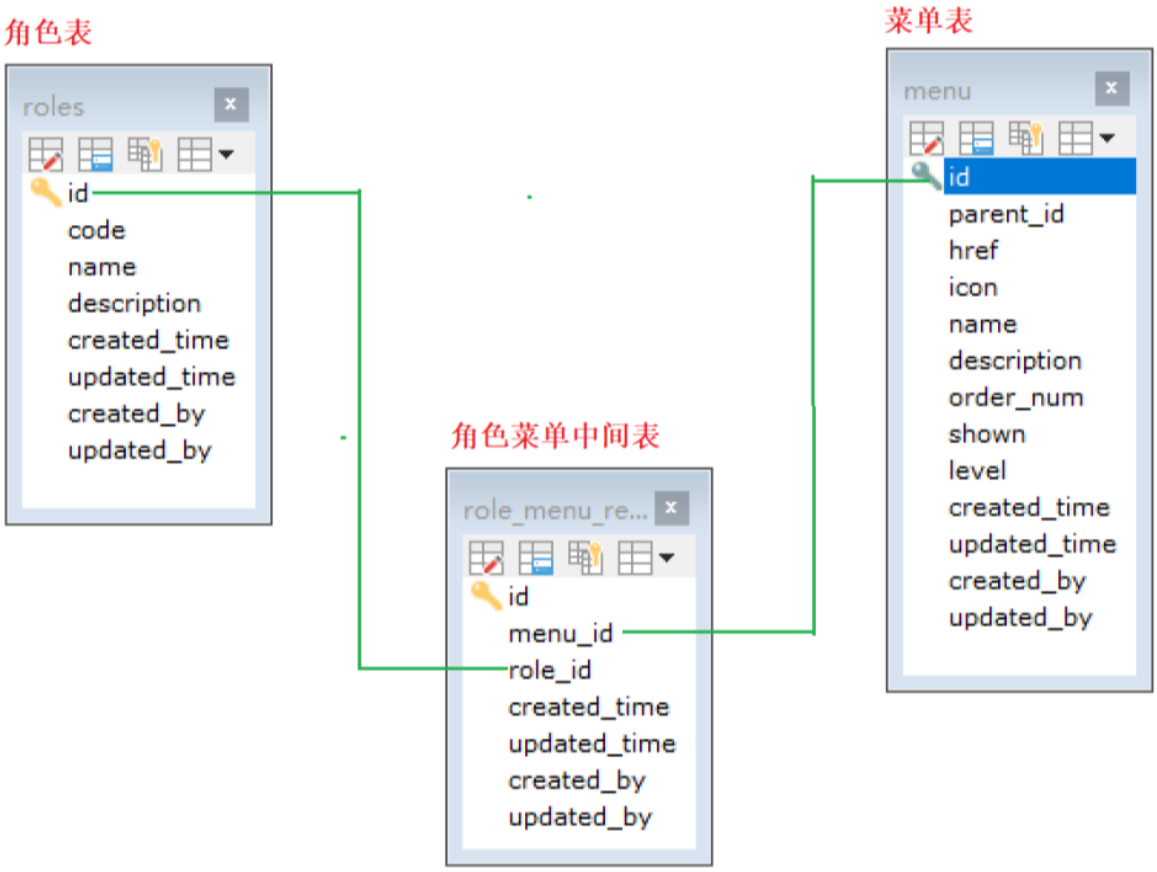


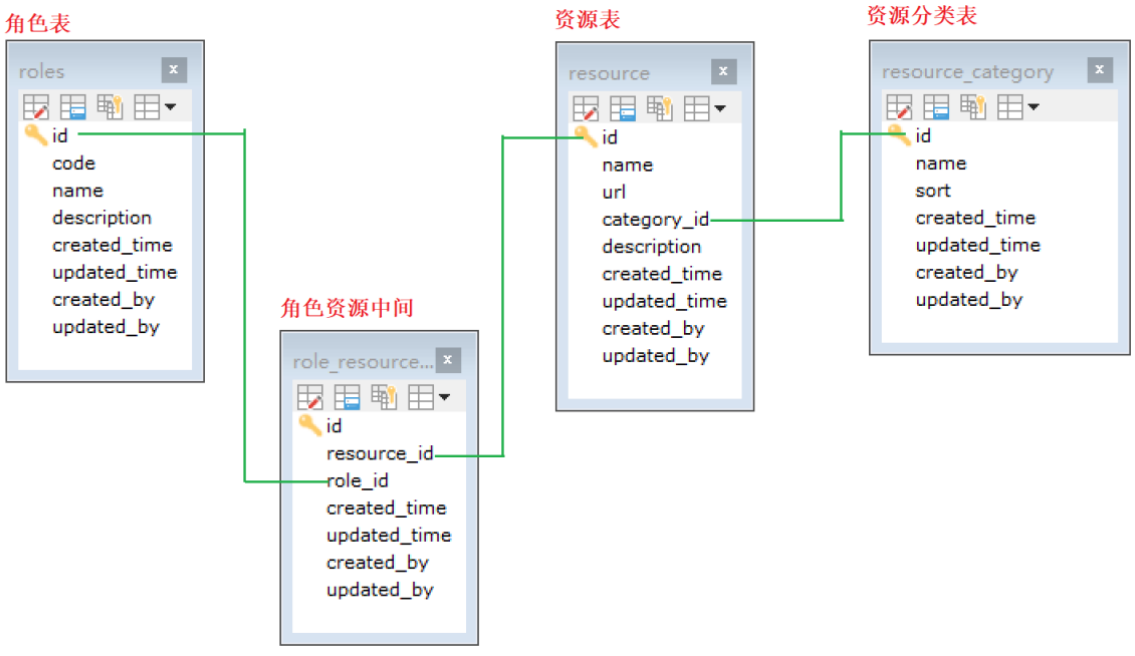
3.2 表关系介绍

1.ER图









2.数据实体描述

2.1 菜单表(menu)

字段类型	类型	约束	描述
id	int(11)	PK	自增主键
parent_id	int(11)		父菜单id,顶级菜父菜单id为-1
href	varchar(200)		菜单路径
icon	varchar(200)		菜单图标
name	varchar(200)		菜单名称
description	varchar(500)		描述
order_num	int(11)		排序号
shown	tinyint(2)		是否显示
level	int(11)		菜单层级，从0开始，越大层级越低
created_time	datetime		创建时间
updated_time	datetime		更新时间
created_by	varchar(100)		创建人
updated_by	varchar(100)		更新人

2.2 资源分类表(resource_category)

字段类型	类型	约束	描述
id	int(11)	PK	自增主键
name	varchar(200)		资源分类名称
sort	int(11)		排序，从小到大顺序排
created_time	datetime		创建时间
updated_time	datetime		更新时间
created_by	varchar(100)		创建人
updated_by	varchar(100)		更新人

2.3 资源表(resource)

字段类型	类型	约束	描述
id	int(11)	PK	自增主键
name	varchar(200)		资源名称
url	varchar(200)		资源url
category_id	int(11)		资源分类ID
description	varchar(500)		描述
created_time	datetime		创建时间
updated_time	datetime		更新时间
created_by	varchar(100)		创建人
updated_by	varchar(100)		更新人

2.4 角色表(roles)

字段类型	类型	约束	描述
id	int(11)	PK	自增主键
code	varchar(100)		角色code
name	varchar(200)		角色名称
description	varchar(500)		描述
created_time	datetime		创建时间
updated_time	datetime		更新时间
created_by	varchar(100)		创建人
updated_by	varchar(100)		更新人

2.5 用户-角色关系表(user_role_relation)

字段类型	类型	约束	描述
id	int(11)	PK	自增主键
user_id	int(11)		用户ID
role_id	int(11)		角色ID
created_time	datetime		创建时间
updated_time	datetime		更新时间
created_by	varchar(100)		创建人
updated_by	varchar(100)		更新人

2.6 角色-菜单关系表(role_menu_relation)

字段类型	类型	约束	描述
id	int(11)	PK	自增主键
menu_id	int(11)		菜单ID
role_id	int(11)		角色ID
created_time	datetime		创建时间
updated_time	datetime		更新时间
created_by	varchar(100)		创建人
updated_by	varchar(100)		更新人

2.7 角色-资源关系表(role_resource_relation)

字段类型	类型	约束	描述
id	int(11)	PK	自增主键
resource_id	int(11)		资源ID
role_id	int(11)		角色ID
created_time	datetime		创建时间
updated_time	datetime		更新时间
created_by	varchar(100)		创建人
updated_by	varchar(100)		更新人

四 权限管理（角色模块）接口实现

1. 角色列表查询&条件查询

1.1 需求分析

需求：点击角色列表按钮进行角色列表展示

1.2 查看接口文档，进行编码

查看接口文档

实体类：Role

```
public class Role {

    private Integer id;
    private String code;
    private String name;
    private String description;
    private Date createdTime;
    private Date updatedTime;
    private String createdBy;
    private String updatedBy;

    //getter/setter...

}
```

Dao层：RoleMapper

```
public interface RoleMapper {

    /*
        查询角色列表(条件)
     */
    public List<Role> findAllRole(Role role);
}
```

```
<?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.RoleMapper">

    <!--查询所有角色（条件）-->
    <select id="findAllRole" resultType="com.lagou.domain.Role">
        SELECT
            id,
            CODE,
            NAME,
            description,
            created_time,
            updated_time,
            created_by,
            updated_by
        FROM roles
        <where>
            <if test="name != null and name != ''">
               and name = #{name}
            </if>
        </where>

    </select>
```

Service层：RoleService

```
public interface RoleService {

    public List<Role> findAllRole(Role role);
}
```

```
@Service
public class RoleServiceImpl implements RoleService {

    @Autowired
    private RoleMapper roleMapper;

    @Override
    public List<Role> findAllRole(Role role) {

        List<Role> allRole = roleMapper.findAllRole(role);
        return allRole;
    }
 }
```

Web层：RoleController

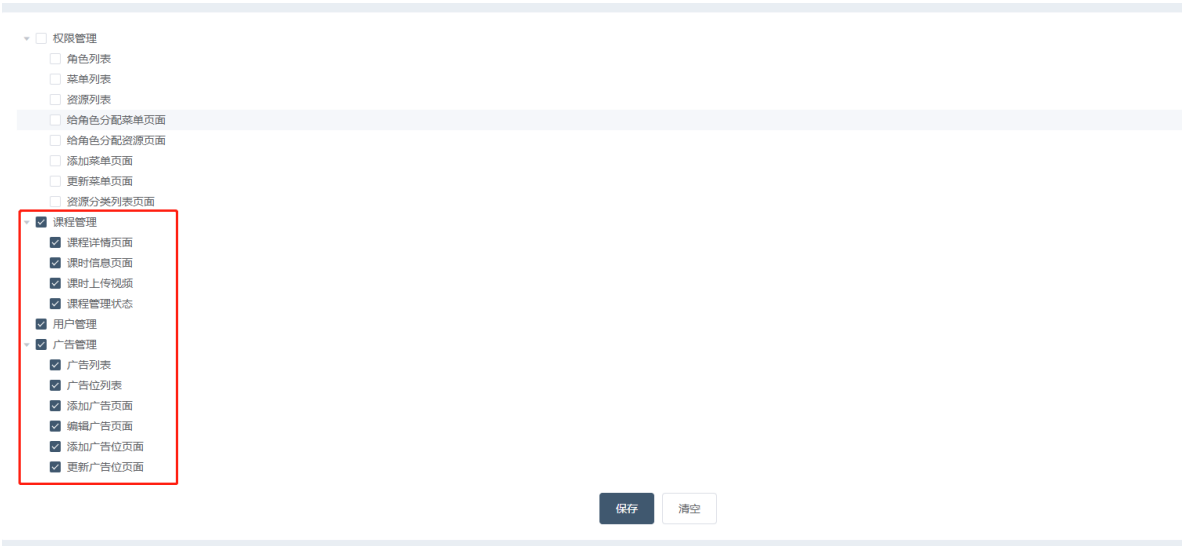
```
@RestController
@RequestMapping("/role")
public class RoleController {

    @Autowired
    private RoleService roleService;

    @RequestMapping("/findAllRole")
    public ResponseResult findAllUserByPage(@RequestBody Role role){

        List<Role> allRole = roleService.findAllRole(role);

        ResponseResult responseResult = new ResponseResult(true,200,"响应成
功",allRole);
        return responseResult;
    }
}
```

1.3 Postman测试接口

2. 分配菜单

2.1 需求分析

需求：点击分配菜单，回显可选择菜单信息，并回显选中状态



2.2 接口1 查询所有菜单列表

查看接口文档

Dao层：MenuMapper

```
public interface MenuMapper {


    /**
     * 查询全部的父子菜单信息
     * */
    public List<Menu> findSubMenuListByPid(int pid);

}
```

```
 <!-- 一对多: 查找子孙菜单 -->
    <select id="findSubMenuListByPid" resultMap="MenuResult">
        select * from menu where parent_id = #{pid}
    </select>

    <!--  根据pid 查询所有子分类集合  -->
    <resultMap id="MenuResult" type="com.lagou.domain.Menu">
        <id column="id" property="id"></id>
        <result column="href" property="href"></result>
        <result column="icon" property="icon"></result>
        <result column="name" property="name"></result>
        <result column="parent_id" property="parentId"></result>
        <result column="description" property="description"></result>
```

```
        <result column="orderNum" property="order_num"></result>
        <result column="shown" property="shown"></result>
        <result column="created_time" property="createdTime"></result>
        <result column="updated_time" property="updatedTime"></result>
        <result column="created_by" property="createdBy"></result>
        <result column="updated_by" property="updatedBy"></result>

        <collection property="subMenuList" ofType="com.lagou.domain.Menu"
                    select="findSubMenuListByPid" column="id" ></collection>
    </resultMap>
```

Service层：MenuService

```
public interface MenuService {

    public List<Menu> findSubMenuListByPid(int pid);
}
```

```
@Service
public class MenuServiceImpl implements MenuService {

    @Autowired
    private MenuMapper menuMapper;

    @Override
    public List<Menu> findSubMenuListByPid(int pid) {
        List<Menu> menuList = menuMapper.findSubMenuListByPid(pid);
        return menuList;
    }

 }
```

Web层：RoleController

```
@RestController
@RequestMapping("/role")
public class RoleController {

    @Autowired
    private MenuService menuService;

/*
        查询所有菜单信息
    */
    @RequestMapping("/findAllMenu")
    public ResponseResult findAllMenu(){
        //-1 表示查询所有菜单数据
        List<Menu> menuList = menuService.findSubMenuListByPid(-1);

        Map<String,Object> map = new HashMap<>();
        map.put("parentMenuList",menuList);
```

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

2.3 Postman测试接口

2.3 接口2 根据角色ID查询关联菜单ID

Dao层：RoleMapperr

```
public interface RoleMapper {

    /*
        根据角色ID查询菜单信息
     */
    List<String> findMenuByRoleId(Integer roleId);
}
```

```
<mapper namespace="com.lagou.dao.RoleMapper">

    <!--  List<String> findMenuByRoleId(Integer roleId);-->
    <select id="findMenuByRoleId" parameterType="int" resultType="string">
        SELECT m.`id`FROM roles r LEFT JOIN role_menu_relation  rm ON r.id =
rm.`role_id`LEFT JOIN menu m ON rm.`menu_id` = m.`id` WHERE r.id = #{id}
    </select>
```

Service层：RoleService

```
public interface RoleService {

        /**
          * 根据ID查询角色关联菜单
         * */
    List<String> findMenuByRoleId(Integer roleId);
}
```

```
@Service
public class RoleServiceImpl implements RoleService {

    @Autowired
    private RoleMapper roleMapper;

    @Override
    public List<String> findMenuByRoleId(Integer roleId) {
        List<String> list = roleMapper.findMenuByRoleId(roleId);
        return list;
    }
 }
```

Web层：RoleController

```
    /**
     * 查询角色关联菜单列表ID
     * */
    @RequestMapping("/findMenuByRoleId")
    public ResponseResult findMenuByRoleId(Integer roleId){
        List<String> menuList = roleService.findMenuByRoleId(roleId);

        ResponseResult result = new ResponseResult(true,200,"响应成功",menuList);
        return result;
    }
```

2.3 Postman测试接口

2.4 接口3 为角色分配菜单列表

Dao层：RoleMapper

```
public interface RoleMapper {

    /*
        角色菜单关联
     */
    void RoleContextMenu(Role_menu_relation role_menu_relation);


}
```

```
<?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.RoleMapper">

    <!--删除角色菜单关联信息-->
```

```
    <delete id="deleteRoleContextMenu" parameterType="int">
        delete from role_menu_relation where role_id = #{id}
    </delete>

     <!--角色菜单关联-->
    <insert id="RoleContextMenu"
parameterType="com.lagou.domain.Role_menu_relation">
        insert into role_menu_relation values(null,#{menuId},#{roleId},#
{createdTime},#{updatedTime},#{createdBy},#{updatedby})
    </insert>


</mapper>
```

Service层：RoleService

```
public interface RoleService {

    void RoleContextMenu(RoleMenuVo roleMenuVo);
}
```

```
@Service
public class RoleServiceImpl implements RoleService {

    @Override
    public void RoleContextMenu(RoleMenuVo roleMenuVo) {

        // 清空中间表
        roleMapper.deleteRoleContextMenu(roleMenuVo.getRoleId());

        for (Integer mid : roleMenuVo.getMenuIdList()) {
            Role_menu_relation role_menu_relation = new Role_menu_relation();
            role_menu_relation.setRoleId(roleMenuVo.getRoleId());
            role_menu_relation.setMenuId(mid);
            role_menu_relation.setCreatedTime(new Date());
            role_menu_relation.setUpdatedTime(new Date());
            role_menu_relation.setCreatedBy("system");
            role_menu_relation.setUpdatedby("system");
            roleMapper.RoleContextMenu(role_menu_relation);
        }

    }


 }
```

Web层：RoleController

```
@RestController
@RequestMapping("/role")
public class RoleController {
```

```
    @Autowired
    private RoleService roleService;

    /*
        用户关联菜单 {roleId: 4, menuIdList: [19, 20, 7, 8, 9, 15, 16, 17, 18]}
     */
    @RequestMapping("/RoleContextMenu")
    public ResponseResult RoleContextMenu(@RequestBody RoleMenuVo roleMenuVo){
        roleService.RoleContextMenu(roleMenuVo);
        ResponseResult result = new ResponseResult(true,200,"响应成功","");
        return result;
    }


}
```

2.3 Postman测试接口

3. 删除角色

3.1 需求分析

需求：点击删除按钮，将选中的角色信息删除

5	普通用户	普通用户只有查看权限	2020-08-17 15:05:40	分配菜单 分配资源 编辑 删除
4	广告管理员	管理广告、广告位信息	2020-07-20 15:55:56	分配菜单 分配资源 编辑 删除
3	课程管理员	管理课程信息，对课程、课时、章节进行管理。	2020-07-20 15:52:04	分配菜单 分配资源 编辑 删除
2	权限管理员	管理权限相关数据，如角色、菜单、资源。可以给用户分配角色。	2020-07-20 15:47:55	分配菜单 分配资源 编辑 删除

3.2 查看接口文档，进行编码

查看接口文档

Dao层：RoleMapper

```
public interface RoleMapper {


    /*
        删除角色
     */
    void deleteRole(Integer id);

}
```

```
<?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.RoleMapper">

      <delete id="deleteRole" parameterType="int">
        delete from roles where id = #{id}
    </delete>
</mapper>
```

Service层：RoleService

```
public interface RoleService {

    void deleteRole(Integer id);
}
```

```
@Service
public class RoleServiceImpl implements RoleService {

    @Autowired
    private RoleMapper roleMapper;

    @Override
    public void deleteRole(Integer id) {
         // 清空中间表
        roleMapper.deleteRoleContextMenu(id);
        roleMapper.deleteRole(id);
    }



 }
```

Web层：RoleController

```
@RestController
@RequestMapping("/role")
public class RoleController {

    @Autowired
    private RoleService roleService;

     /**
     * 删除角色
     * */
    @RequestMapping("/deleteRole")
    public ResponseResult deleteRole(Integer id){

        roleService.deleteRole(id);
```

```
        ResponseResult responseResult = new ResponseResult(true,200,"响应成
功","");
        return responseResult;
    }

}
```

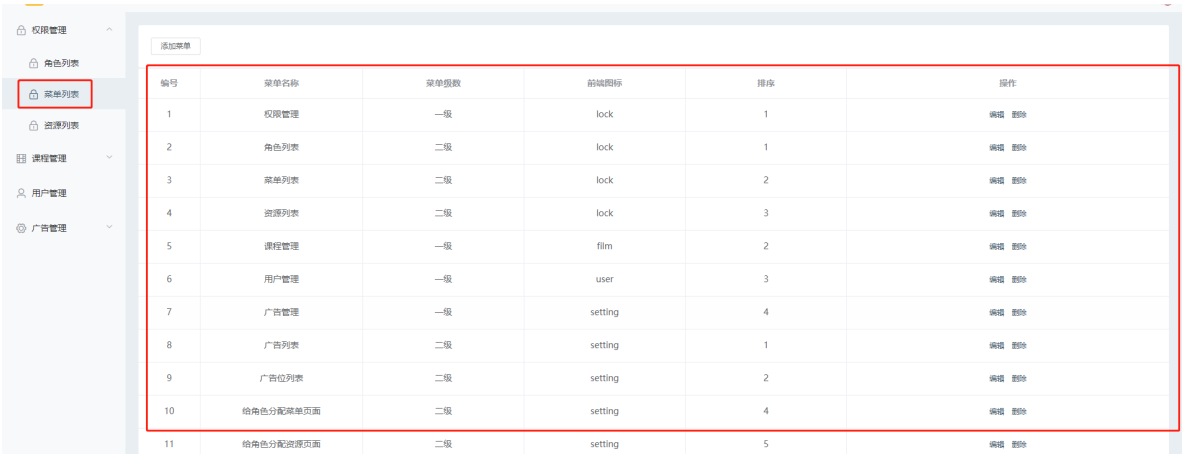
3.3 Postman测试接口

五 权限管理（菜单模块）接口实现

1. 菜单列表查询

1.1 需求分析

需求：点击菜单列表，对菜单信息进行列表展示



1.2 查看接口文档，进行编码

查看接口文档

实体类：Menu

```
public class Menu {

       //主键id
    private Integer id;

    //父菜单id
    private int parentId;

    //菜单路径
    private String href;

    //菜单图标
    private String icon;

    //菜单名称
    private String name;
```

```
    //描述
    private String description;

    //排序号
    private int orderNum;

    //是否展示
    private int shown;

    //菜单层级，从0开始
    private int level;

    //创建时间
    private Date createdTime;

    //更新时间
    private Date updatedTime;

    //创建人
    private String createdBy;

    //更新人
    private String updatedBy;

    // getter/setter..
}
```

Dao层：MenuMapper

```
public interface MenuMapper {

    /**
     * 查询菜单列表
     * */
    public List<Menu> findAllMenu();


}
```

```
<mapper namespace="com.lagou.dao.MenuMapper">

    <!--  查询菜单列表  -->
    <select id="findAllMenu" resultType="com.lagou.domain.Menu">
        SELECT
            id,
            parent_id,
            href,
            icon,
            NAME,
            description,
            order_num,
            shown,
            LEVEL,
            created_time,
            updated_time,
```

```
            created_by,
            updated_by
        FROM menu
    </select>

</mapper>
```

Service层：MenuService

```
public interface MenuService {

    public List<Menu> findAllMenu();
}
```

```
@Service
public class MenuServiceImpl implements MenuService {

    @Autowired
    private MenuMapper menuMapper;

    @Override
    public List<Menu> findAllMenu() {

        List<Menu> list = menuMapper.findAllMenu();

        return list;
    }
```

Web层：MenuController

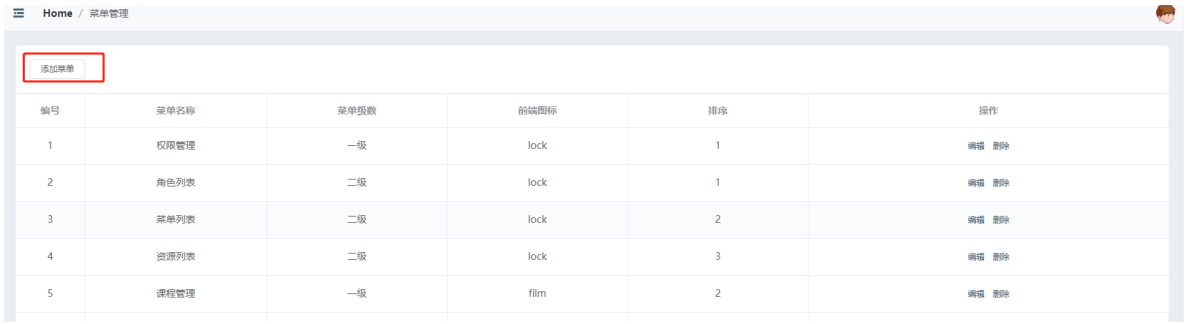
```
@RestController
@RequestMapping("/menu")
public class MenuController {

    @Autowired
    private MenuService menuService;

    /**
     * 查询菜单列表信息
     * */
    @RequestMapping("/findAllMenu")
    public ResponseResult findAllMenu(){

        List<Menu> list = menuService.findAllMenu();
        ResponseResult result = new ResponseResult(true,200,"响应成功",list);

        return result;
    }


}
```

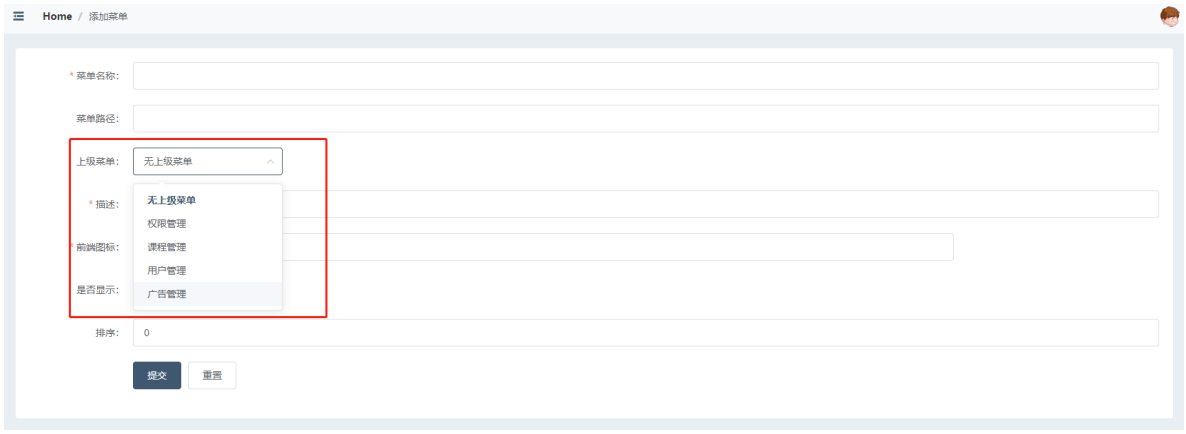
Postman测试接口

2. 查询菜单信息(回显)

2.1 需求分析

需求：点击添加菜单按钮，跳转到添加菜单页面，回显当前添加菜单可以选择的上级菜单信息





2.2 查看接口文档，进行编码

查看接口文档

Dao层：MenuMapper

```
public interface MenuMapper {

 /**
     * 查询全部的父子菜单信息
     * */
    public List<Menu> findSubMenuListByPid(int pid);


}
```

```
<mapper namespace="com.lagou.dao.MenuMapper">

  <!-- 一对多: 查找子孙菜单 -->
```

```
    <select id="findSubMenuListByPid" resultMap="MenuResult">
        select * from menu where parent_id = #{pid}
    </select>

    <!--  根据pid 查询所有子分类集合  -->
    <resultMap id="MenuResult" type="com.lagou.domain.Menu">
        <id column="id" property="id"></id>
        <result column="href" property="href"></result>
        <result column="icon" property="icon"></result>
        <result column="name" property="name"></result>
        <result column="parent_id" property="parentId"></result>
        <result column="description" property="description"></result>
        <result column="orderNum" property="order_num"></result>
        <result column="shown" property="shown"></result>
        <result column="created_time" property="createdTime"></result>
        <result column="updated_time" property="updatedTime"></result>
        <result column="created_by" property="createdBy"></result>
        <result column="updated_by" property="updatedBy"></result>

        <collection property="subMenuList" ofType="com.lagou.domain.Menu"
                    select="findSubMenuListByPid" column="id" ></collection>
    </resultMap>

</mapper>
```

Service层：MenuService

```
public interface MenuService {

    public List<Menu> findSubMenuListByPid(int pid);
}
```

```
@Service
public class MenuServiceImpl implements MenuService {

    @Autowired
    private MenuMapper menuMapper;

  @Override
    public List<Menu> findSubMenuListByPid(int pid) {
        List<Menu> menuList = menuMapper.findSubMenuListByPid(pid);
        return menuList;
    }
}
```

Web层：MenuController

```
@RestController
@RequestMapping("/menu")
public class MenuController {

    @Autowired
```

```
    private MenuService menuService;

   /**
     * 回显菜单信息(包括父子菜单的全部信息)
     * */
    @RequestMapping("/findMenuInfoById")
    public ResponseResult findMenuInfoById(@RequestParam int id){

        if(id == -1){
            //添加操作 回显不需要查询 menu信息
            List<Menu> menuList = menuService.findSubMenuListByPid(-1);

            //封装数据
            Map<String,Object> map = new HashMap<>();
            map.put("menuInfo",null);
            map.put("parentMenuList",menuList);

            ResponseResult result = new ResponseResult(true,200,"响应成功",map);
            return result;
        }else{
            //修改操作 回显
            Menu menu = menuService.findMenuById(id);
            List<Menu> menuList = menuService.findSubMenuListByPid(-1);

            Map<String,Object> map = new HashMap<>();
            map.put("menuInfo",menu);
            map.put("parentMenuList",menuList);

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

    }

}
```

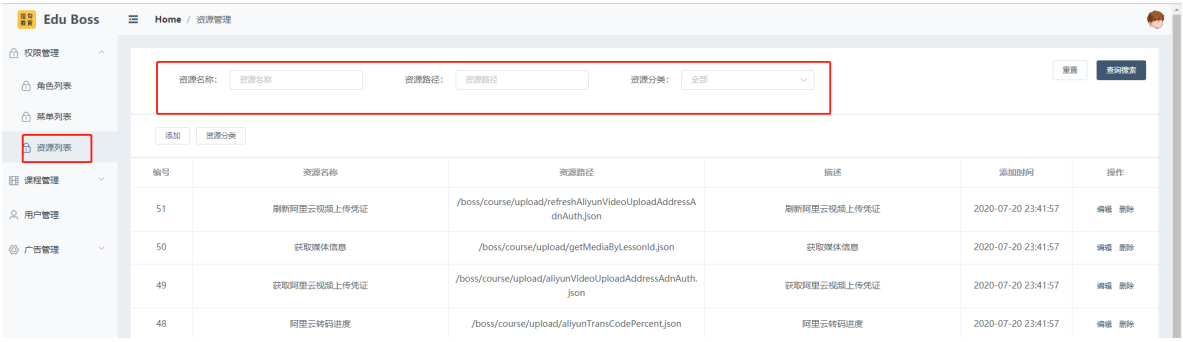
2.3 Postman测试接口

六 权限管理（资源模块）接口实现

1.资源分页&多条件查询

1.1 需求分析

需求：资源列表及多条件组合查询



1.2 查看接口文档，进行编码

查看接口文档

Dao层：ResourceMapper

```
public interface ResourceMapper {

    public List<Resource> findAllResource(ResourceVo resourceVo);
}
```

```
<?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.ResourceMapper">

    <!--查询所有资源-->
    <select id="findAllResource" resultType="com.lagou.domain.Resource">
        select * from resource
        <where>

            <if test="name != null">
                and name like  concat('%',#{name},'%')
            </if>

            <if test="url != null">
                and url = url
            </if>

            <if test="categoryId != null">
                and category_id = #{categoryId}
            </if>

        </where>

    </select>
</mapper>
```

Service层：ResourceService

```
public interface ResourceService {

    public PageInfo<Resource> findAllResource(ResourceVo resourceVo);
}
```

```
@Service
public class ResourceServiceImpl implements ResourceService {

    @Autowired
    private ResourceMapper resourceMapper;

    @Override
    public PageInfo<Resource> findAllResource(ResourceVo resourceVo) {


 PageHelper.startPage(resourceVo.getCurrentPage(),resourceVo.getPageSize());
        List<Resource> allResource = resourceMapper.findAllResource(resourceVo);

        PageInfo<Resource> adPageInfo = new PageInfo<Resource>(allResource);

        return adPageInfo;
    }
```

Web层：ResourceController

```
@RestController
@RequestMapping("/resource")
public class ResourceController {

    @Autowired
    private ResourceService resourceService;

    /**
     * 分页与条件查询
     * */
    @RequestMapping("/findAllResource")
    public ResponseResult findAllResource(@RequestBody ResourceVo resourceVo){

        PageInfo<Resource> allResource =
resourceService.findAllResource(resourceVo);

        ResponseResult responseResult = new ResponseResult(true,200,"响应成
功",allResource);
        return responseResult;

    }

}
```

1.3 Postman测试接口

七 登陆及动态菜单展示

1. 登陆

1.1 需求分析

需求：输入用户名密码，点击登陆按钮，进行用户登陆

加密算法MD5介绍

1、什么是MD5

MD5加密全程是Message-Digest Algoorithm 5（信息-摘要算法），它对信息进行摘要采集，再通过一定的位运算，最终获取加密后的MD5字符串。

2、MD5有哪些特点

MD5加密的特点主要有以下几点：

1、针对不同长度待加密的数据、字符串等等，其都可以返回一个固定长度的MD5加密字符串。（通常32位的16进制字符串）；

2、其加密过程几乎不可逆，除非维护一个庞大的Key-Value数据库来进行碰撞破解，否则几乎无法解开。

3、运算简便，且可实现方式多样，通过一定的处理方式也可以避免碰撞算法的破解。（加盐：随机字符串）

4、对于一个固定的字符串。数字等等，MD5加密后的字符串是固定的，也就是说不管MD5加密多少次，都是同样的结果。

3、Java代码中如何使用MD5

（1）添加依赖

```
<!--MD5依赖-->
        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-lang3</artifactId>
            <version>3.3.2</version>
        </dependency>
        <dependency>
            <groupId>commons-codec</groupId>
            <artifactId>commons-codec</artifactId>
            <version>1.3</version>
        </dependency>
```

（2）添加工具类

```
public class Md5 {

    public final static  String md5key = "Ms2";
    /**
     * MD5方法
     * @param text 明文
     * @param key 密钥
     * @return 密文
     * @throws Exception
     */
    public static String md5(String text, String key) throws Exception {
        //加密后的字符串
        String encodeStr= DigestUtils.md5Hex(text+key);
        System.out.println("MD5加密后的字符串为:encodeStr="+encodeStr);
        return encodeStr;
    }

    /**
     * MD5验证方法
     * @param text 明文
     * @param key 密钥
     * @param md5 密文
     * @return true/false
     * @throws Exception
     */
    public static boolean verify(String text, String key, String md5) throws
Exception {
        //根据传入的密钥进行验证
        String md5Text = md5(text, key);
        if(md5Text.equalsIgnoreCase(md5))
        {
            System.out.println("MD5验证通过");
            return true;
        }
        return false;
```

```
    }

    public static void main(String[] args) throws Exception {
        // 注册  用户名：tom  密码  123456
        // 添加用户的时候，要进行加密
        String lagou = Md5.md5("123456", "lagou");
        System.out.println(lagou);

        // 登陆  用户名 tom   123456 select * from user where username = tom and
password = 123456
        // 1.根据用户名进行查询 f00485441dfb815c75a13f3c3389c0b9

        boolean verify = Md5.verify("123456", "lagou",
"f00485441dfb815c75a13f3c3389c0b9");
        System.out.println(verify);


    }

}
```

1.2 查看接口文档，进行编码

查看接口文档

Dao层：UserMapper

```
public interface UserMapper  {

    /*
        用户登陆
     */
    public User login(User user);
}
```

```
<?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.UserMapper">

  <!-- 用户登陆 -->
    <select id="login" parameterType="com.lagou.domain.User"
resultType="com.lagou.domain.User">
            select * from user where phone = #{phone}
    </select>
</mapper>
```

Service层：UserService

```
public interface UserService {

    /*
         用户登录
     */
     public User login(User user);
}
```

```
@Service
public class UserServiceImpl implements UserService {

    @Autowired
    private UserMapper userMapper;

    /**
    * 用户登录
     * */
        @Override
    public User login(User user) throws Exception {

        User user2 = userMapper.login(user);

        if(user2 != null &&
Md5.verify(user.getPassword(),"lagou",user2.getPassword())){
            return user2;
        }else {
            return null;
        }


    }

 }
```

Web层：UserController

```
@RestController
@RequestMapping("/user")
public class UserController {

    @Autowired
    private UserService userService;

        /**
     * 用户登录
     * */
    @RequestMapping("/login")
    public ResponseResult login(User user, HttpServletRequest request) throws
Exception {

        User login = userService.login(user);

        ResponseResult result = null;
        if(login !=null ){
```

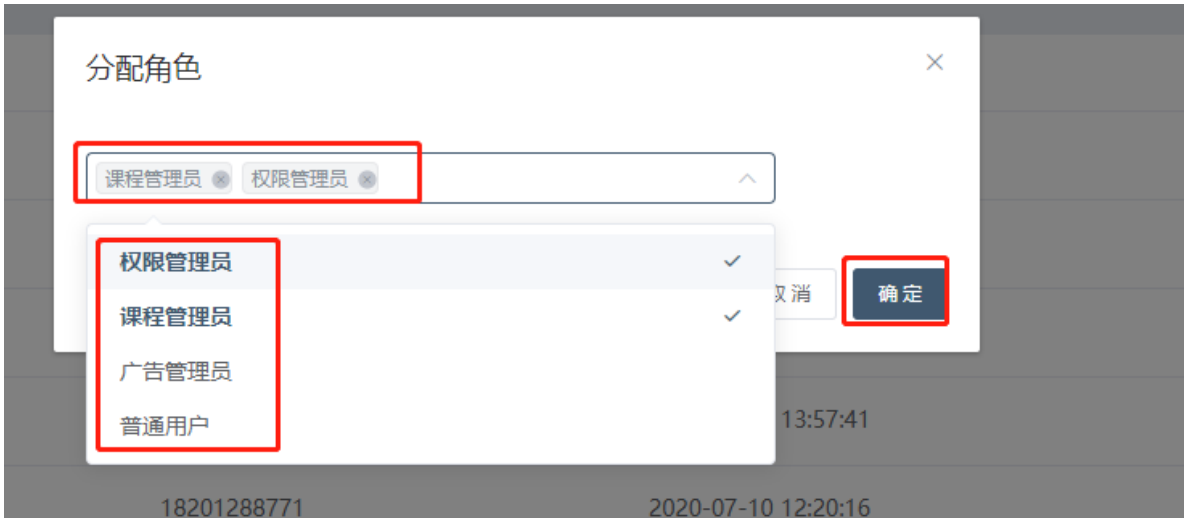
```
            //保存access_token
            Map<String,Object> map = new HashMap<>();
            String  access_token = UUID.randomUUID().toString();
            map.put("access_token", access_token);
            map.put("user_id",login.getId());

            HttpSession session = request.getSession();
            session.setAttribute("user_id",login.getId());
            session.setAttribute("access_token",access_token);

             result = new ResponseResult(true,1,"响应成功",map);
        }else{
             result = new ResponseResult(true,1,"用户名密码错误",null);
        }


        return result;
    }

}
```

1.3 Postman测试接口

2. 分配角色（回显）

1.1 需求分析

需求：点击分配角色，将该用户所具有的角色信息进行回显



1.2 查看接口文档，进行编码

查看接口文档

Dao层：UserMapper

```
public interface UserMapper {

    /**
     * 根据ID查询用户当前角色
     * */
    public List<Role> findUserRelationRoleById(int id);
}
```

```
<?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.UserMapper">
     <!-- 根据ID查询用户当前角色    -->
    <select id="findUserRelationRoleById" resultType="com.lagou.domain.Role"
parameterType="int">
        SELECT
            r.id,
            r.code,
            r.name,
            r.description
        FROM roles r INNER JOIN user_role_relation ur
        ON r.`id` = ur.`role_id` INNER JOIN USER u ON ur.`user_id` = u.`id`
        WHERE u.`id` = #{id}
    </select>

</mapper>
```

Service层：UserService

```
public interface UserService {

    /**
     * 获取用户拥有的角色
     * */
    public List<Role> findUserRelationRoleById(int id) ;
}
```

```
@Service
public class UserServiceImpl implements UserService {

    @Autowired
    private UserMapper userMapper;

    /**
     * 获取用户拥有的角色
     * */
    @Override
    public List<Role> findUserRelationRoleById(int id) {
```

```
        List<Role> roleList = userMapper.findUserRelationRoleById(id);
        return roleList;
    }


}
```

Web层：UserController

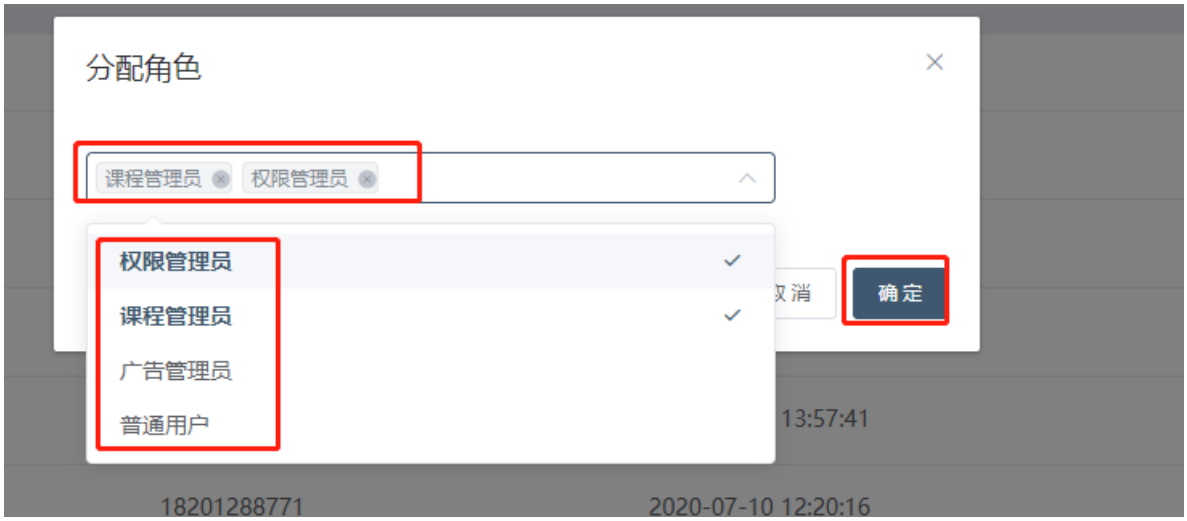
```
@RestController
@RequestMapping("/user")
public class UserController {

     @Autowired
    private UserService userService;

    /*
    获取用户拥有的角色
     */
    @RequestMapping("/findUserRoleById")
    public ResponseResult findUserRoleById(int id){
        List<Role> roleList = userService.findUserRelationRoleById(id);
        return  new ResponseResult(true,200,"分配角色回显成功",roleList);
    }

}
```

1.3 Postman测试接口

3. 分配角色

1.1 需求分析

需求：点击确定按钮，真正实现用户角色关联



1.2 查看接口文档，进行编码

查看接口文档

Dao层：UserMapper

```
public interface UserMapper {

       /*
        根据用户ID清空中间表
     */
    void deleteUserContextRole(Integer userId);


    /*
        分配角色
     */
    void userContextRole(User_Role_relation user_role_relation);
}
```

```
<?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.UserMapper">

    <!-- 根据userid清空中间表关联关系     void deleteUserContextRole(Integer
userId);-->
    <delete id="deleteUserContextRole" parameterType="int">
        delete from  user_role_relation where user_id = #{userid}
    </delete>

    <!--用户角色关联     void userContextRole(Integer userId, Integer roleid);-->
    <insert id="userContextRole"
 parameterType="com.lagou.domain.User_Role_relation">
        insert into user_role_relation values(null,#{userId},#{roleId},#
{createdTime},#{updatedTime},#{createdBy},#{updatedby})
    </insert>
</mapper>
```

Service层：UserService

```
public interface UserService {

       /*
         用户关联角色
        */
    void userContextRole(UserVo userVo);
}
```

```
@Service
public class UserServiceImpl implements UserService {

    @Autowired
    private UserMapper userMapper;

    /*
        用户关联角色
     */
    @Override
    public void userContextRole(UserVo userVo) {

        // 根据用户ID清空中间表的关联关系
        userMapper.deleteUserContextRole(userVo.getUserId());

        // 向中间表添加记录
        for (Integer roleid : userVo.getRoleIdList()) {

            User_Role_relation user_role_relation = new User_Role_relation();
            user_role_relation.setUserId(userVo.getUserId());
            user_role_relation.setRoleId(roleid);
            Date date = new Date();
            user_role_relation.setCreatedTime(date);
            user_role_relation.setUpdatedTime(date);

            user_role_relation.setCreatedBy("system");
            user_role_relation.setUpdatedby("system");

            userMapper.userContextRole(user_role_relation);
        }
    }


 }
```

Web层：UserController

```
@RestController
@RequestMapping("/user")
public class UserController {

     @Autowired
```

```
    private UserService userService;

       /*
            分配角色
       */
    @RequestMapping("/userContextRole")
    public ResponseResult userContextRole(@RequestBody UserVo userVo){

        userService.userContextRole(userVo);

        return  new ResponseResult(true,200,"分配角色成功",null);
    }

}
```

1.3 Postman测试接口

4. 动态菜单显示

1.1 需求分析

需求：登陆成功后，根据用户所拥有的权限信息，进行菜单列表动态展示

1.2 查看接口文档，进行编码

查看接口文档

Dao层：UserMapper

```
public interface UserMapper {

    /**
     * 根据ID查询用户当前角色
```

```
     * */
    public List<Role> findUserRelationRoleById(int id);

    /**
     * 根据角色id,查询角色拥有的顶级菜单信息
     * */
    public List<Menu> findParentMenuByRoleId(List<Integer> ids);

    /**
     * 根据PID 查询子菜单信息
     * */
    public List<Menu> findSubMenuByPid(int pid);

    /**
     * 获取用户拥有的资源权限信息
     * */
    public List<Resource> findResourceByRoleId(List<Integer> ids);

}
```

```
<?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.UserMapper">

     <!-- 根据ID查询用户当前角色   -->
    <select id="findUserRelationRoleById" resultType="com.lagou.domain.Role"
parameterType="int">
        SELECT
            r.id,
            r.code,
            r.name,
            r.description
        FROM roles r INNER JOIN user_role_relation ur
        ON r.`id` = ur.`role_id` INNER JOIN USER u ON ur.`user_id` = u.`id`
        WHERE u.`id` = #{id}
    </select>

    <!--  根据角色id,查询角色拥有的顶级菜单信息  -->
    <select id="findParentMenuByRoleId" parameterType="java.util.List"
resultType="com.lagou.domain.Menu">
          SELECT
             DISTINCT  m.*
         FROM roles r INNER JOIN role_menu_relation rm ON r.`id` = rm.role_id
                INNER JOIN menu m ON rm.menu_id = m.id
                 WHERE m.parent_id = -1 AND r.id IN
           <foreach  collection="list" item="item"  open="(" separator=","
close=")">
                #{item}
         </foreach>
          GROUP BY m.id
    </select>


    <!-- 根据PID 查找子菜单 -->
    <select id="findSubMenuByPid" resultType="com.lagou.domain.Menu">
```

```
        select * from menu where parent_id = #{pid}
    </select>


    <!--  获取用户拥有的资源权限  -->
    <select id="findResourceByRoleId" parameterType="java.util.List"
resultType="com.lagou.domain.Resource">
                SELECT
          DISTINCT rc.*
        FROM roles r  INNER JOIN role_resource_relation rrr ON r.`id` =
rrr.`role_id`
        INNER JOIN resource rc ON rrr.`resource_id` = rc.`id` WHERE r.id IN
        <foreach item="item" index="index" collection="list" open="("
separator="," close=")">
            #{item}
        </foreach>
        GROUP BY rc.id;
    </select>
</mapper>
```

Service层：UserService

```
public interface UserService {

          /*
            * 获取用户权限
         * */
       ResponseResult getUserPermissions(Integer id);
}
```

```
@Service
public class UserServiceImpl implements PromotionSpaceService {

   @Autowired
    private UserMapper userMapper;


    @Override
    public ResponseResult getUserPermissions(Integer id) {

        //1.获取当前用户拥有的角色
        List<Role> roleList = userMapper.findUserRelationRoleById(id);

        //2.获取角色ID,保存到 list
        List<Integer> list = new ArrayList<>();

        for (Role role : roleList) {
            list.add(role.getId());
        }

        //3.根据角色id查询 父菜单
        List<Menu> parentMenu = userMapper.findParentMenuByRoleId(list);


        //4.封装父菜单下的子菜单
```

```
        for (Menu menu : parentMenu) {
            List<Menu> subMenu = userMapper.findSubMenuByPid(menu.getId());
            menu.setSubMenuList(subMenu);
        }

        //5.获取资源权限
        List<Resource> resourceList = userMapper.findResourceByRoleId(list);


        //6.封装数据
        Map<String,Object> map = new HashMap<>();
        map.put("menuList",parentMenu);  //menuList: 菜单权限数据
        map.put("resourceList",resourceList);//resourceList: 资源权限数据

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


}
```

Web层：UserController

```
@RestController
@RequestMapping("/user")
public class UserController {

     @Autowired
    private UserService userService;

    /**
     * 获取用户权限
     * */
    @RequestMapping("/getUserPermissions")
    public ResponseResult getUserPermissions(HttpServletRequest request){

        //获取请求头中的 token
        String token = request.getHeader("Authorization");

        //获取session中的access_token
        HttpSession session = request.getSession();
        String access_token = (String)session.getAttribute("access_token");

        //判断
        if(token.equals(access_token)){
            int user_id = (Integer)session.getAttribute("user_id");
            ResponseResult result = userService.getUserPermissions(user_id);
            return result;
        }else{
            ResponseResult result = new ResponseResult(false,400,"获取失败","");
            return result;
        }
    }
}
```

1.3 Postman测试接口