类型管理

1.类型管理需求

1.类型的查询

2.根据类型的名称模糊查询

3.类型的添加

4.批量删除类型

5.单个删除类型

2.后台提供的相关接口

2.1 model 相关类

```
@Data
public class Type implements Serializable {
    private Integer id;
    private String typeName;

//    public Integer getId() {
//        return id;
//    }
//
//    public void setId(Integer id) {
//        this.id = id;
//    }
//
//    public String getTypeName() {
//        return typeName;
//    }
//
//    public void setTypeName(String typeName) {
//        this.typeName = typeName;
//    }
}
```

2.2 dao 层接口和实现(mapper 实现)

接口

```
@Component("typeDao")
public interface TypeMapper {

    /**
```

```
     * 查询 高级查询
     */
    List<Type> listType(@Param("typeName") String typeName);

    /**
     * 添加
     */
    int insertType(Type type);

    /**
     * 删除 根据 id 删除
     */
    int deleteTypeByID(Integer id);

}
```

实现层

```
<mapper namespace="com.zhanghk.dao.TypeMapper">
    <select id="listType" parameterType="string" resultType="com.zhangh
k.model.Type">
        select id,type_name from type
        <where>
            <if test="typeName!=null and typeName!=''">
                and type_name like '%${typeName}%'
            </if>
        </where>
    </select>

    <insert id="insertType" parameterType="com.zhanghk.model.Type">
        insert insert type(type_name) values(#{typeName})
    </insert>

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

2.3 Service 层接口和实现

接口

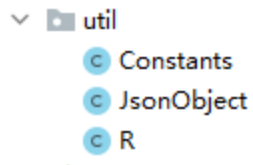
```
public interface TypeService {

    /**
     * 分页查询
     */
    PageInfo<Type> findAll(int page,int pageSize,String typeName);
```

```
    /**
     * 添加
     */
    int insertType(Type type);

    /**
     * 删除
     */
    int deleteTypeByID(Integer id);

}
```

实现

```
@Service("typeService")
public class TypeServiceImpl implements TypeService {

    @Autowired
    private TypeMapper typeDao;

    @Override
    public PageInfo<Type> findAll(int page, int pageSize, String typeNa
me) {
        PageHelper.startPage(page,pageSize);
        List<Type> list = typeDao.queryList(typeName);
        PageInfo<Type> pageInfo = new PageInfo<>(list);
        return pageInfo;
    }

    @Override
    public int insertType(Type type) {
        return typeDao.insertType(type);
    }

    @Override
    public int deleteTypeByID(Integer id) {
        return typeDao.deleteTypeByID(id);
    }
}
```

2.4 工具类的应用



2.5 Controller 层

```
@RestController
@RequestMapping("/type")
public class TypeController {

    @Autowired
    private TypeService typeService;

    /**
     * 类型查询接口
     */
    @RequestMapping("/queryAll")
    public JsonObject queryAll(@RequestParam(defaultValue = "1") Integer page,
                               @RequestParam(defaultValue = "10") Integer pageSize,
                               String typeName){
        //创建返回的对象
        JsonObject object = new JsonObject();
        //查询类型列表信息
        PageInfo<Type> pageInfo = typeService.findAll(page,pageSize,typeName);
        //填充返回的前端数据对象
        object.setCode(0);
        object.setMsg("ok");
        object.setCount(pageInfo.getTotal());
        object.setData(pageInfo.getList());

        return object;
    }
```

```
    /**
     * 添加类型接口
     */
    @RequestMapping("/add")
    public R add(@RequestBody Type type){
        int flag = typeService.insertType(type);
        if (flag>0){
            return R.ok();
        }
        return R.fail("添加类型失败");
    }
    /**
     * 类型删除接口，支持单个删除和批量删除
     */
    @RequestMapping("/deleteByIds")
    public R deleteByIds(String ids){//批量删除，前台可以传入“1，2，3，4”这样的 id 组
        //将传入的 id 组字符串转为集合对象
        List<String> list =  Arrays.asList(ids.split(","));
        //遍历
        for (String id:list) {
            typeService.deleteTypeByID(Integer.parseInt(id));
        }
        return R.ok();
    }
}
```

3.前端接口的调用

3.1 查询接口的调用

```
table.render({
    elem: '#currentTableId',
    url: 'http://127.0.0.1:8888/type/queryAll', //数据信息
    toolbar: '#toolbarDemo',
    defaultToolbar: ['filter', 'exports', 'print', {
        title: '提示',
        layEvent: 'LAYTABLE_TIPS',
        icon: 'layui-icon-tips'
    }],
    cols: [[
        {type: "checkbox", width: 50},
        {field: 'id', width: 120, title: 'ID', sort: true},
        {field: 'typeName', width: 150, title: '类型名称'},
        {title: '操作', minWidth: 300, toolbar: '#currentTableBar', align: "center"}
    ]],
```

3.2 高级查询

页面处理

```
<legend>根据类型名称查询</legend>
<div style="margin: 10px 10px 10px 10px">
    <form class="layui-form layui-form-pane" action="">
        <div class="layui-form-item">
            <div class="layui-inline">
                <label class="layui-form-label">类型名称</label>
                <div class="layui-input-inline">
                    <input type="text" name="typeName" autocomplete="off" class="layui-input">
                </div>
            </div>

            <div class="layui-inline">
                <button type="submit" class="layui-btn layui-btn-primary"  lay-submit lay-filter="data-search-btn"><i class=
            </div>
        </div>
    </form>
</div>
```

js 的调用

两个 typeName 绑定

```
// 监听搜索操作
form.on('submit(data-search-btn)', function (data) {
    var result = JSON.stringify(data.field);
    var typeName = data.field.typeName;
    layer.alert(result, {
        title: '最终的搜索信息'
    });

    //执行搜索重载
    table.reload('currentTableId', {
        page: {
            curr: 1
        }
        , where: {
            typeName: typeName
        }
    }, 'data');

    return false;
});
```

```
/**
 * 类型查询接口
 */
@RequestMapping("/queryAll")
public JsonObject queryAll(@RequestParam(defaultValue = "1") Integer page,
                           @RequestParam(defaultValue = "10") Integer pageSize,
                           String typeName){
    //创建返回的对象
```

3.3 类型的添加功能（axios 异步添加）

步骤：

1. 点击打开添加页面 admin/page/type/add.html

```
<div class="layui-form layuimini-form">
    <div class="layui-form-item">
        <label class="layui-form-label required">类型名称</label>
        <div class="layui-input-block">
            <input type="text" name="typeName" lay-verify="required" lay-reqtext="类型名不能为空" placeholder="请输入类型名" value="'
            <tip>填写丢失物品的类型的名称。</tip>
        </div>
    </div>


    <div class="layui-form-item">
        <div class="layui-input-block">
            <button class="layui-btn layui-btn-normal" lay-submit lay-filter="saveBtn">确认保存</button>
        </div>
    </div>
</div>
```

2. 输入类型名称，点击确认保存按钮，提交到后台 （跨域）

```
//监听提交
form.on('submit(saveBtn)', function (data) {
    //先获取从Layui点击后的信息;
    var datas = data.field;
    //获取信息后，并发送信息到后端服务器（以往习惯使用Ajax发送，现在用axois处理,其实就是Ajax的二次封装）
    axios.post('http://127.0.0.1:8888/type/add',datas).then(function (response) {
        if(response.code==200){
            //添加成功
            layer.msg("添加成功");
            //刷新页面 整个主页面都会刷新
            window.reload();

        }
        // 关闭弹出层
        layer.close(parentIndex);
    }).catch(function (error) {
        layer.msg(data.msg)
    })
```

3. 服务器端处理完成后返回状态

```
    */
    @RequestMapping("/add")
    public R add(@RequestBody Type type){
        int flag = typeService.insertType(type);
        if (flag>0){
            return R.ok();
        }
        return R.fail("添加类型失败");
    }
```

4. 前端接收状态刷新主页

```
axios.post('http://127.0.0.1:8888/type/add',datas).then(function (response)
    if(response.code==200){
        //添加成功
        layer.msg("添加成功");
        //刷新页面 整个主页面都会刷新
        window.reload();

    }
    // 关闭弹出层
```

```
84
85          //前台输入数据，后台添加后，刷新页面会整体，但这里只让table刷新
86          window.reload = function () {
87              table.reload("currentTableId")
88          }
```

bug 记录：

【1】：axios 使用错误

```
<script>
    layui.use(['form', 'table','axios'], function () {
        var form = layui.form,
            layer = layui.layer,
            table = layui.table,
            axios = layui.axios, //刚开始这个地方写成layer.axios了，出现问题，没反应。
            $ = layui.$;

        /**
```

【2】：mybatis 中的 mapper 文件中 sql 语法错误

```
<insert id="insertType" parameterType="com.zhanghk.model.Type">
    insert insert type(type_name) values(#{typeName})
</insert>
```

正确写法

```
insert into type(type_name)values(#{typeName})
```

3.4 类型的删除 –批量删除、单个删除

步骤：

1、 判断是否有选中的记录信息，如果没有进行提示

2、 获取要删除的 id 集合

3、 发送删除请求，实现删除功能

4、 根据返回结果提示，并且刷新 table 局部