service和element-ui

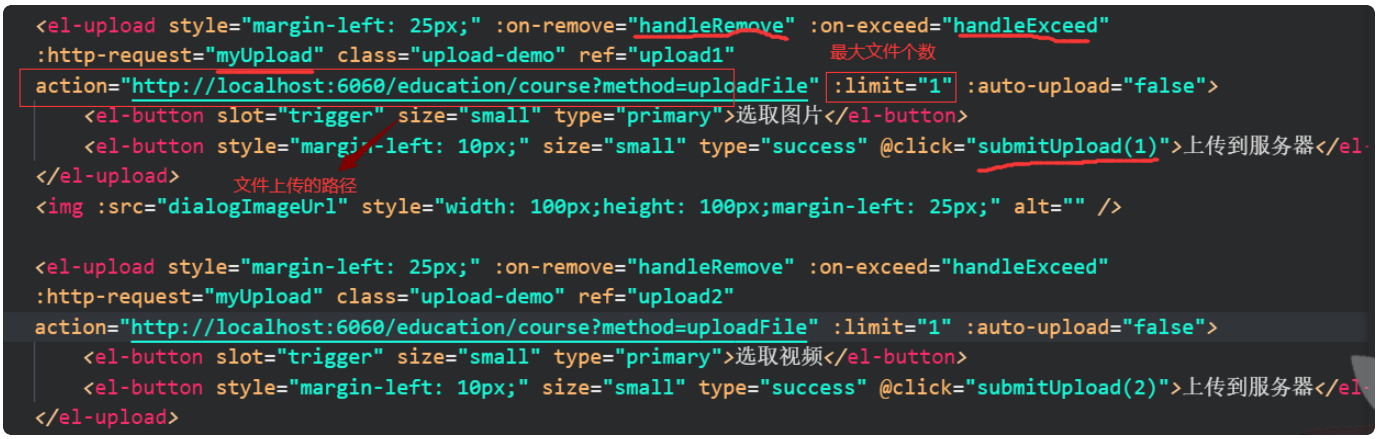
1.角色表设计

- 用户和管理员都是一张表，通过属性区分，先通过普通用户名和密码查询，再根据它的角色属性不同，跳转不同的首页进行控制即可.

2.验证码功能

- 只有ajax,axios发送请求才有跨域
- session获取验证码失败，axios跨服务器访问获取不到 ---使用servletContext获取
- 前端校验验证码，在后台保存到变量中，在前台请求这个变量获取即可，在登录请求发送之前获取验证么信息即可,前端校验

3.elementui文件上传




```
<el-upload style="margin-left: 25px;" :on-remove="handleRemove" :on-exceed="handleExceed"
:http-request="myUpload" class="upload-demo" ref="upload1"
action="http://localhost:6060/education/course?method=uploadFile" :limit="1" :auto-upload="false">
    <el-button slot="trigger" size="small" type="primary">选取图片</el-button>
    <el-button style="margin-left: 10px;" size="small" type="success" @click="submitUpload(1)">上传到服务器</el
</el-upload>
<img :src="dialogImageUrl" style="width: 100px;height: 100px;margin-left: 25px;" alt="" />

<el-upload style="margin-left: 25px;" :on-remove="handleRemove" :on-exceed="handleExceed"
:http-request="myUpload" class="upload-demo" ref="upload2"
action="http://localhost:6060/education/course?method=uploadFile" :limit="1" :auto-upload="false">
    <el-button slot="trigger" size="small" type="primary">选取视频</el-button>
    <el-button style="margin-left: 10px;" size="small" type="success" @click="submitUpload(2)">上传到服务器</el
</el-upload>
```

handleExceed(files,fileList)	文件个数上线触发；files代表本次选择几个文件，fileList代表已经上传多少个文件
handleRemove(file,fileList)	点击文件错号触发,file代表的是你要删除的文件
submitUpload("1")	点击上传服务器的时候触发，传参数是为了标识那个组件提交
myUpload(content)	在我们点击submitUpload的时候，this.$refs.upload1.submit()上传的时候触发，content.file是一个文件对象，我们可以在后台直接使用part获取.可以通过content.action获取上传服务器路径

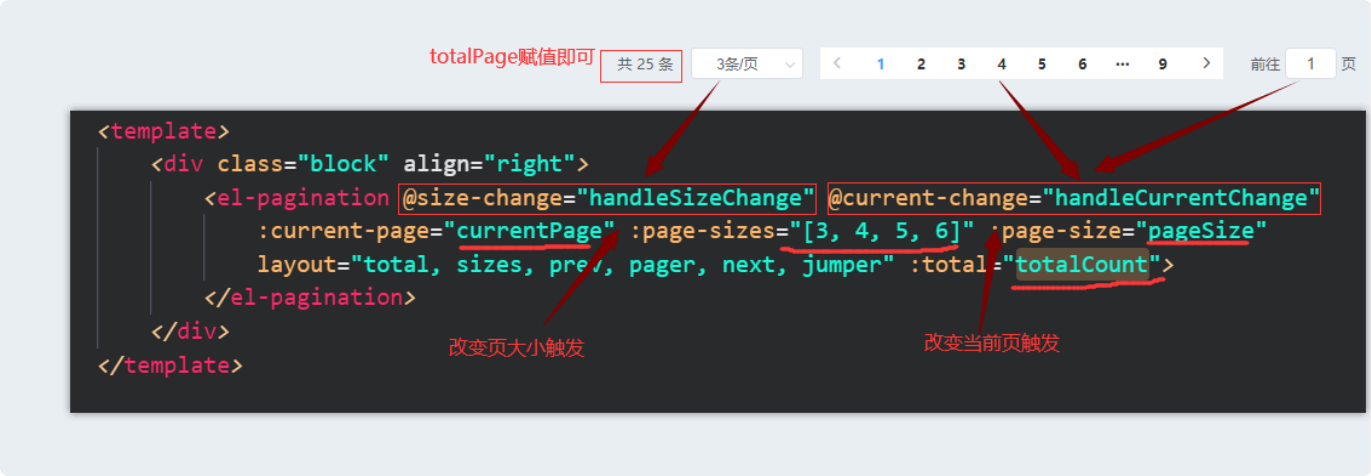
- limit-----限制个数,
- ref-----用于标识组件
- action------上传服务器路径

上传文件发送异步请求的特殊性:post fromdata

JavaScript | 复制代码

```javascript
//正常发送格式
let params = new FormData();
params.append('method', "updateBook");
params.append('bid', this.bid);
params.append('bookname', this.book.bookname);
axios.post(this.url, params)
  .then((res) => {
  console.log(res.data);
});
 //我们点击服务器按钮触发
 submitUpload(value) {
      if (value == 1) {
          //文件1提交
           this.$refs.upload1.submit();
           this.flagImage = true;
     } else {
           this.$refs.upload2.submit();
           this.flagVideo = true;
     }
 },
//文件上传最好这么发送
 myUpload(content) {
 //发送fomData请求---二进制上传
let formData = new FormData();
formData.append("file", content.file);
 axios({
       method: "post", //请求方式
       url: content.action, //请求路径
       data: formData //数据
 }).then(response => {
        console.log(response.data);
        if (response.data.code == 201) {
              this.dialogImageUrl = response.data.data;
        } else {
              this.dialogVedioUrl = response.data.data;
              console.log(this.dialogVedioUrl);
        }
});
```

4.element-ui分页查询



handleSizeChange:function (size) {}	这个函数在我们改变页大小触发，size就是用户要求的每页几条数据
handleCurrentChange:function (currentPage) {}	这个函数在改变当前页的时候触发,currentPage代表当前页，需要向后台传输的数据
selectAll()	可以自定义一个函数，用于查询所有，前两个函数直接调用即可

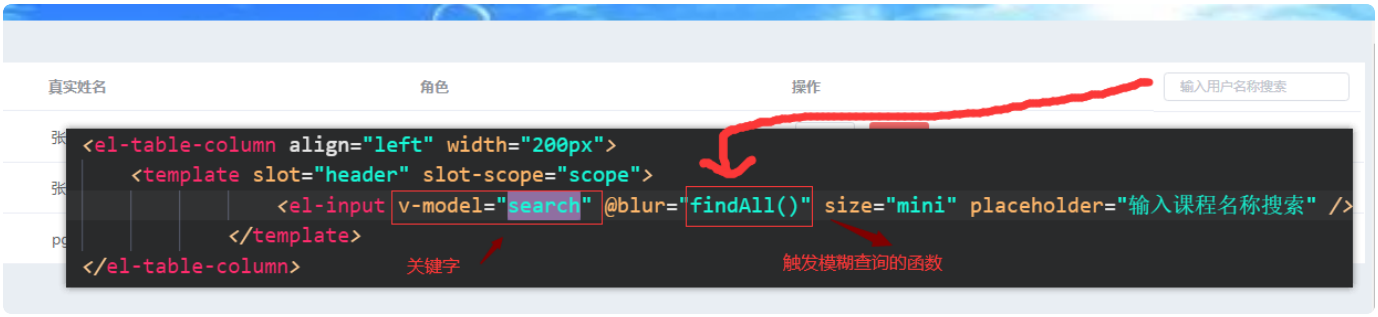
- currentPage--------当前页：用户改变双向绑定的；
- pageSize---------每页多少条：用户改变双向绑定的;
- totalCount-------总条数，后台传递过来的，我们需要给它赋值，全查条数，模糊条件的条数
- totalPage--------总页数，后台传递过来的，我们需要给它赋值，总页数，后台计算出来的

分页查询代码创建的时机模糊带分页

Java 复制代码

```java
/**
* 用户模块---分页+模糊 http://localhost:6060/education/user?method=selectUserAll
* @param request
* @param response
*/
private void selectUserAll(HttpServletRequest request,HttpServletResponse response) throws IOException {
    //模糊查询
    String search = request.getParameter("search");
    //默认查询---分页查询
    String pageSize1 = request.getParameter("pageSize");
    int pageSize=Integer.parseInt(request.getParameter("pageSize"));
    int totalCount=userService.selectTotalCount(search);
    String currentPage=request.getParameter("currentPage");
    PageUtils pageUtils = new PageUtils(pageSize,totalCount,currentPage);
    //查询所有
    List<User> userList=userService.selectUserAll(pageUtils.getStartIndex(),pageUtils.getPageSize(),search);
    ResultVo vo=null;
    Map<String,Object> map=new HashMap<>();
    map.put("userList",userList);
    map.put("pageUtils",pageUtils);
    if (userList!=null) {
        vo=new ResultVo(200,"查询成功",map);
    }else {
        vo=new ResultVo(500,"查询失败",null);
    }
    response.getWriter().write(JSONUtil.toJsonStr(vo));
    }
```

5.element-ui模糊查询



模糊查询的sql

```java
    public List<User> selectUserAll(int startIndex, int pageSize, String search) {
        String sql="select * from user where 1=1";
        StringBuilder builder = new StringBuilder(sql);
        if(search!=null&&!"".equals(search)){
            builder.append(" and username like '%"+search+"%'");
        }
        builder.append(" limit ?,?");
        List<User> userList=null;
        try {
            userList=queryRunner.query(builder.toString(),new BeanListHandler<>(User.class),startIndex,pageSize);
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
        return userList;
    }
```

6.element-ui对话框

修改和详情触发的函数，点击就触发

7.element-ui表单验证

Form 组件提供了表单验证的功能，只需要通过 `rules` 属性传入约定的验证规则，并将 Form-Item 的 `prop` 属性设置为需校验的字段名即可。校验规则参见 async-validator

```
<el-form :model="ruleForm" :rules="rules" ref="ruleForm" label-width="100px" class="demo-ruleForm">
  <el-form-item label="活动名称" prop="name">
    <el-input v-model="ruleForm.name"></el-input>
  </el-form-item>
```

```
<div slot="footer" class="dialog-footer" style="margin-left: 20px;">
    <el-button @click="handleClose">取 消</el-button>
    <el-button type="primary" @click="submitForm2('ruleForm')">确 定</el-button>
</div>
```

```
methods: {
  submitForm(formName) {
    this.$refs[formName].validate((valid) => {
      if (valid) {
        alert('submit!');
      } else {
        console.log('error submit!!');
        return false;
      }
    });
  },
  resetForm(formName) {
    this.$refs[formName].resetFields();
  }
```

表单提交就会触发它

valid:代表表单校验是否成功，如果成功，他就会是true

如果校验失败，他就是false

8.element-ui表格数据渲染（详情、修改）

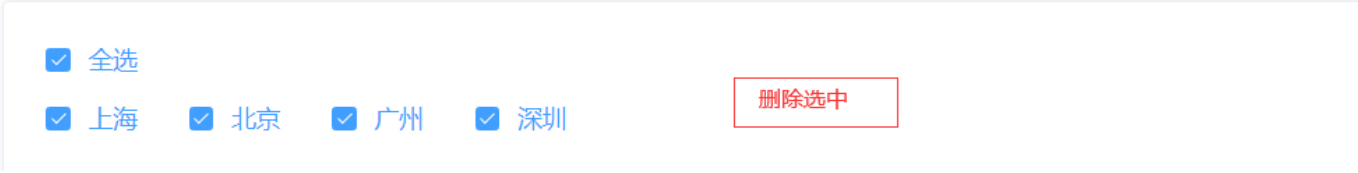
```
<template slot-scope="scope">
    <el-button size="mini" @click="handleEdit(scope.$index, scope.row)">修改</el-button>
    <el-button size="mini" type="danger" @click="handleLook(scope.$index, scope.row)">详情
    </el-button>
</template>
```

```
methods: {
  handleEdit(index, row) {
    console.log(index, row);
  },
  handleDelete(index, row) {
    console.log(index, row);
  }
}
```

我们必须在这里将对话框标志改为true

index 代表第几条数据， row代表整条数据对象

9.element-ui删除选中



Java | 复制代码

```
handleCheckAllChange(val) {
  //这个函数在点击单选的时候触发  val代表选中复选框的数组
},
handleCheckedCitiesChange() {
  //删除选中的时候触发
}
```