小U课堂笔记

1.验证码实现

2.登录业务实现

3.token校验(前后端)

4.退出按钮(session.invalidate)

5.用户管理模块

  5.1展示用户数据(所有)

  5.2分页展示

6.模糊查询

7.增删改查

8.文件的上传

1.验证码实现

借助验证码工具类,生成四位随机数，存入session中,(用于后期比对校验)生成验证码图片,输出流写到前端,为了保证实时刷新，我们只需改变路径即可，但是浏览器一般不识别同样的路径，我们需要给一个随机参数，保证浏览器再次发送请求。注意新版本google不让携带cookie了(需要其他方法解决,url路径拼接)

2.登录业务实现

前端传参数：用户名密码验证码，应该先进行验证码比对，然后通过校验用户是否存在，如果用户存在，在比对密码

```
        //校验都通过了，发送登录请求！
        //axios是在Vue管理的页面中发送请求的方式！
        let params = new URLSearchParams();//保存传递参数的对象，打包我们所需要传递的所有数据
        params.append("username", this.ruleForm.username);
        params.append("password", this.ruleForm.password);
        params.append("imageCode", this.ruleForm.imageCode);
        params.append("func", "userLogin");

        axios({
            method: "post",//请求方式
            url: "http://localhost:8080/BjXiaoUManager/user",//请求路径
            data: params
        }).then(response => {
            //对于结果的处理在then方法中编写！

        })
    } else {
        this.$message({
            type: "error",
            message: "校验未通过！"
        })
    }
});
```

3.token校验(前后端)

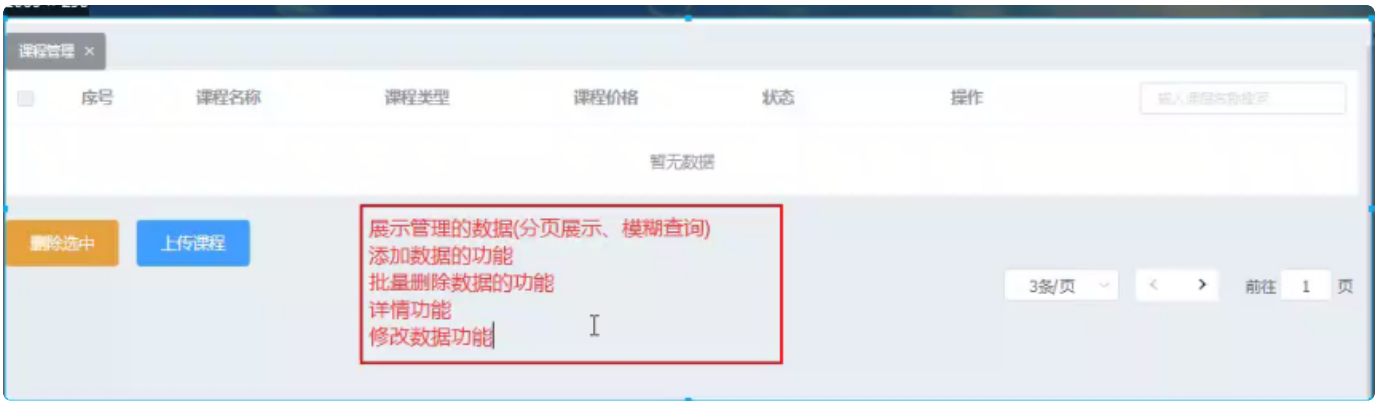
注意：前后端分离的项目，session不在同一个服务器中，session作用范围只用于后台服务器（所以判断未登录要在后台进行逻辑判断或者url传参方式）
在后台写将用户信息存入session中如果不为null，则代表登录过直接响应页面一个json字符串，前台操作json即可

4.退出按钮(session.invalidate)

即销毁了数据，又销毁了会话

```javascript
axios({
    method:"post",
    url:"/login?func=logout",
    data:{
        "":""
    }
}).then(response=>{
    window.location="login.html";
})
```

5.用户管理模块



5.1展示用户数据(所有)

1. response.getWriter().write():写字符串(对象转json---写到前端)
2. created(){} vue的声明周期函数,相当于文档加载函数

JavaScript | 复制代码

```
element-UI:<el-table></el-table>
:data="tableData"   //代表数据来源
//获取所有用户请求
axios({
  method:"get",
  url:"/user?func=findAllUser"
}).then(res=>{
  if(res.data.code==200){
    this.tableData=res.data.data;
  }
})
```

```
    在ElementUI中，表格展示数据，单位是列，不是行！
    label属性 表头的关键字
    prop属性 就是显示的数据关键字
-->
<el-table-column type="selection" width="55" prop="uid">
</el-table-column>
<el-table-column width="100px" label="序号" type="index">
</el-table-column>
<el-table-column label="用户账号" prop="username">
</el-table-column>
<el-table-column label="真实姓名" prop="name">
</el-table-column>
<el-table-column label="角色" prop="role">
    <template slot-scope="{row}">
        <span v-if="row.role==1">管理员</span>
        <span v-if="row.role==2">总经理</span>
        <span v-if="row.role==3">员工</span>
    </template>
</el-table-column>
```

5.2分页展示

分页怎么做：

1. 页面数据：三个参数，总数据量(totalCount)，当前页码(currentPage)，每页多少条(pageSize),直接将这，当前页码，每页多少条两个参数拼接到查询所有传到后台，我们只需计算出总数据量，然后根据当前页码，每页多少条计算出起始id，去查询数据库即可，这样就把查询所有变成了分页查询
2. 分页按钮功能实现：

理解el-pagination:

```
<el-pagination @size-change="handleSizeChange" :slot="tableData"
    @current-change="handleCurrentChange" :current-page="currentPage" :page-sizes="[3, 4, 5, 6]"
    :page-size="pageSize" layout="total, sizes, prev, pager, next, jumper, slot"
    :total="totalCount">
</el-pagination>
```

两个关键数据,要修改的数据,让后带数据重新发送查询数据请求

currentPage	当前页码
pageSize	每页多少条

使用el-pagination:我们也只需要在data中定义三个参数或者四个即可其他是封装好的

6.模糊查询

理解

v-model指令：改变那边都会改变，主要用于表单元素绑定

v-bind指令：单项绑定，主要用于非表单非用户输入的，我们要渲染的

实现方式：将当前页变为第一页：然后分页加模糊重新发送请求。

7.增删改查

JavaScript | 复制代码

```javascript
//表单校验,valid代表校验通过返回true
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();
      }
    }
//vue对象遍历
ruleForm: {
          name: '',
          region: '',
          date1: '',
          date2: '',
          delivery: false,
          type: [],
          resource: '',
          desc: ''
 }
for(let key in ruleForm){
  //key代表name  this.ruleform[key]代表value
}
```

多个数据封装套路：javaweb

```java
/*
    老思路：
        单个获取请求中的参数，有几个获取几个，然后手动构建对象，存储数据！
    新思路：
        使用BeanUtils工具，快速的完成Bean对象的数据封装
        完成赋值的前提条件，参数数据的关键字，必须与JavaBean对象中属性名一致！
 */
Map<String, String[]> map = request.getParameterMap();
User user = new User();
BeanUtils.populate(user, map);
System.out.println(user);
```

注意：element-UI修改和详情可以用对话框组件进行操作

row 代表对象，index代表索引

8.文件的上传

1. element_ui的上传组件

```
    el-upload 是ElementUI中提供的上传组件，借助这个组件，可以完成上传的功能！
    方法
        handleRemove 撤销上传文件时执行，点文件后面那个小叉叉
        handleExceed 当上传的文件超出了组件设定的文件数量时执行
        myUpload 是上传的核心功能代码
        submitUpload 真正触发上传代码执行的方法
    属性
        :limit="1" 限制当前组件上传文件的数量
        ref="upload1"   为当前的上传组件设置一个别名名称
        action 是上传的路径，文件上传，要将文件提交给谁，就是请求谁！
        :auto-upload="false" 取消组件的自动上传
-->
<el-upload style="margin-left: 25px;" :on-remove="handleRemove" :on-exceed="handleExceed"
    :http-request="myUpload" class="upload-demo" ref="upload1"
    action="http://localhost/education/courses?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)">上传至
    </el-button>
</el-upload>
<img :src="dialogImageUrl" style="width: 100px;height: 100px;margin-left: 25px;" alt="" />
```

2. 前端上传核心组件代码编写

```
myUpload(content) {
    /*
        content对象中两个属性极为关键：
            action 上传提交的路径地址
            file 上传时传递的文件，也就是请求发送时的数据
        此时请求发送时，要传递文件对象，文件对象在传递时：
            1、必须采用post请求方式，因为post才能传递二进制数据！
            2、axios构建参数时，不能使用URLSearchParams对象，需要使用FormData对象！
    */
    let params = new FormData();
    params.append("file", content.file);//文件
    params.append("func", "uploadFile");//执行的方法
    axios({
        method: "post",
        url: content.action,
        data: params
    }).then(response => {

    })
},
//触发上传核心方法执行的方法
```

```javascript
methods: {
  submitUpload() {
    this.$refs.upload.submit();
  }
```

3. 发送请求后台接收数据二进制方式：注解形式接收二进制

@Mulitconfig

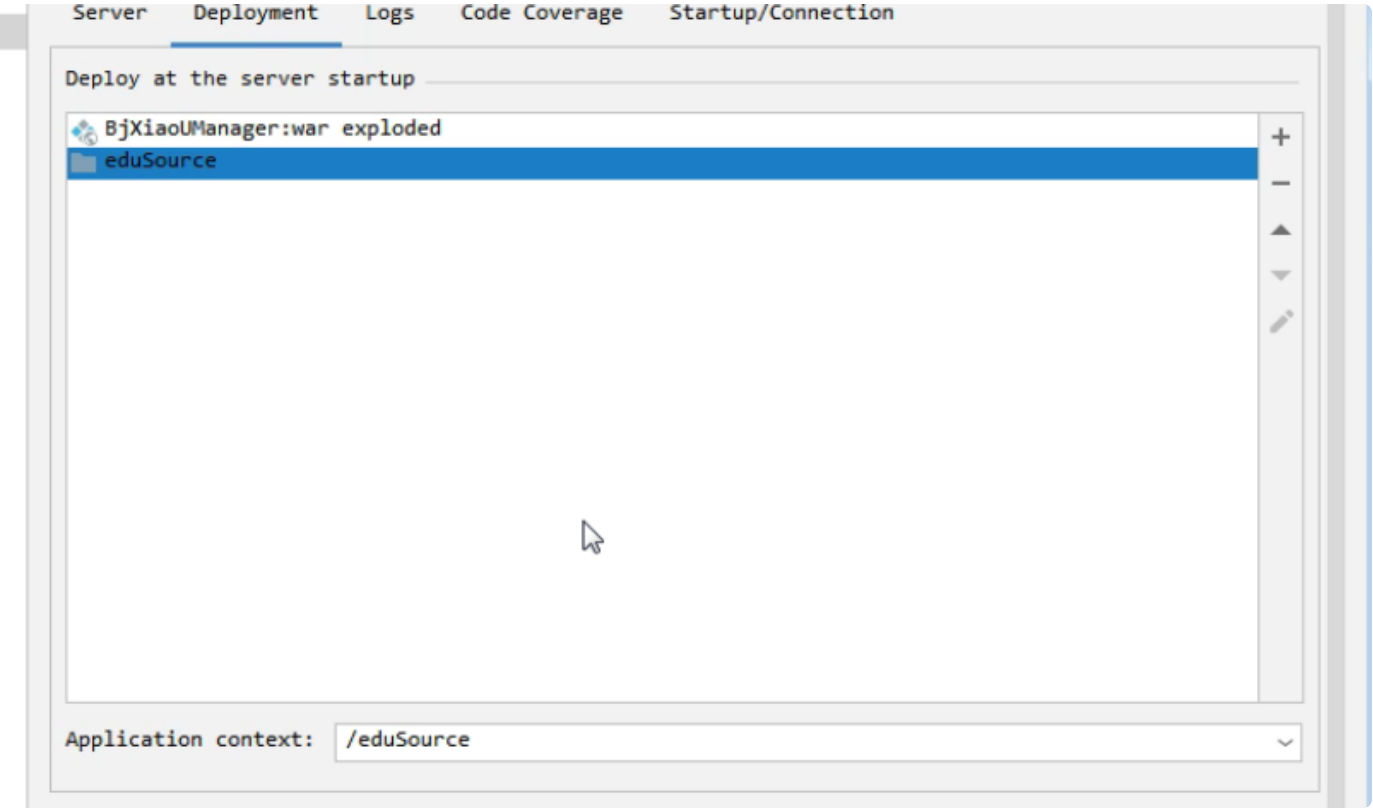
4. 后端上传核心代码

```java
//文件上传的方法
private void uploadFile(HttpServletRequest request, HttpServletResponse response) {
    /*
        此时，拿到传过来的文件！！！
        上传的目的、目标是什么？
            1、获取到上传的文件名
            2、保存上传的文件
     */
    request.getPart( name: "fule")
}
```

```java
    Part对象就代指我们上传的文件！
 */
Part part = request.getPart( name: "file");
//获取上传的文件名
String filename = part.getSubmittedFileName();
//将上传的内容生成一个新的文件保存
String dirPath = "E:/eduSource";
File file = new File(dirPath);
if (!file.exists()) {
    file.mkdirs();//创建文件夹
}
//将上传的内容在文件夹中生成一个新的文件
part.write( fileName: dirPath + "/" + filename);
}
```

保存文件，上传

5. 使用uuid加名字防止重复
6. 回显路径图片url



注意：回显上传只玩转所有的文件名，路径都是拼接的，如果响应结果为两种情况如.jpg/.mp3等，我们只需要借助一个判断标识，响应一个不同状态码，让前端根据不同状态码对url进行赋值

7. elementui文件删除操作

```
//移除上传文件的方法
handleRemove(file, fileList) {
    /*
        该方法中需要做的事情：
        1、清空页面中图片或者视频的显示
        2、将原有上传的文件，从上传文件夹中删除
    */
},
```

```
        2、将原有上传的文件，从上传文件夹中删除
    */
    let delFilename = "";//保存要删除的文件名
    let filename = file.name;//源文件名
    let type = filename.substring(filename.lastIndexOf(".") + 1);//文件类型
    if (type == "jpg" || type == "png" || type == "jpeg" || type == "gif") {
        delFilename =
            this.dialogImageUrl.substring(this.dialogImageUrl.lastIndexOf("/") + 1);
        this.dialogImageUrl = "";//清空图片路径
    } else {
        delFilename =
            this.dialogVedioUrl.substring(this.dialogVedioUrl.lastIndexOf("/") + 1);
        this.dialogVedioUrl = "";//清空视频路径
    }
},
```

```
//发送axios请求到Servlet，对文件进行删除
axios({
    method: "get",
    url: "http://localhost:8080/BjXiaoUManager/course?func=deleteFile&filename=" +
        delFilename
}).then(response => {

})
```