web综合案例

1.登录功能

1.1 登录功能

（1）`MemberServlet`中添加login方法

```
  public Result login(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        Member member = getData(request,Member.class);
        member = memberService.login(member.getEmail(),member.getPassword());

        if(member != null){
            return new Result("登录成功！", member);
        }else{
            return new Result("用户名密码错误，请重试！", false, null,
Code.LOGIN_FAIL);
        }
    }
```

（2）`Code`中添加 常量

```
public static final Integer LOGIN_FAIL = 50101;
```

（3）补全service和impl和dao、xml代码

`MemberService`

```
 /**
     * 根据email和密码登录
     * @param email
     * @param password
     * @return
     */
    Member login(String email, String password);
```

`MemberServiceImpl`

```
 @Override
    public Member login(String email, String password) {
        SqlSession sqlSession = null;
        try{
            //1.获取SqlSession
            sqlSession = MapperFactory.getSqlSession();
            //2.获取Dao
            MemberDao memberDao = MapperFactory.getMapper(sqlSession,
MemberDao.class);
            password = MD5Util.md5(password);
```

```
            Member member = memberDao.findByEmailAndPwd(email,password);
            return member;
        }catch (Exception e){
            e.printStackTrace();
            throw new RuntimeException(e);
            //记录日志
        }finally {
            try {
                TransactionUtil.close(sqlSession);
            }catch (Exception e){
                e.printStackTrace();
            }
        }
    }
```

`MemberDao`

```
Member findByEmailAndPwd(@Param("email") String email,@Param("password") String
password);
```

`MemberDao.xml`

```
<!--配置查询的列名公共SQL语句-->
<sql
id="Base_Column_List">id,nick_name,password,gender,birthday,email,telephone,addr
ess,register_date,state </sql>

<select id="findByEmailAndPwd" parameterType="map" resultMap="BaseResultMap">
    select
    <include refid="Base_Column_List"/>
    from tr_memberwhere email = #{email,jdbcType=VARCHAR} and password = #
{password,jdbcType=VARCHAR}
</select>
```

(4) 修改`login.html`，添加164行左右

```
 //根据返回的结果进行下一步的动作
 if( res.flag){
     // 跳转页面 index.html
     window.open("index.html","_self");
 }else{
    alert(res.message);
 }
```

启动服务器，访问页面，进行登录

1.2 将登陆信息保存到redis中

(1) 将资料中`工程资源文件`中的`JedisUtils`拷贝工程目录下。

(2) 将资料中`工程资源文件`中的`jedis.properties`拷贝到resources下

(3) 将登录用户id信息放入到redis是，修改``login 方法，添加如下代码

```
@Override
    public Member login(String email, String password) {
        SqlSession sqlSession = null;
        try{
            //1.获取SqlSession
            sqlSession = MapperFactory.getSqlSession();
            //2.获取Dao
            MemberDao memberDao = MapperFactory.getMapper(sqlSession,
MemberDao.class);
            password = MD5Util.md5(password);
            Member member = memberDao.findByEmailAndPwd(email,password);

            //3.将登录人的信息保存到redis中
            Jedis jedis = JedisUtils.getResource();
            //使用登录人的id作为key，设定3600秒的过期时间，value值待定
            jedis.setex(member.getId(),3600,"");
            jedis.close();

            return member;
        }catch (Exception e){
            e.printStackTrace();
            throw new RuntimeException(e);
            //记录日志
        }finally {
            try {
                TransactionUtil.close(sqlSession);
            }catch (Exception e){
                e.printStackTrace();
            }
        }
    }
```

启动redis服务器。

启动项目服务进行测试

1.3 登陆状态校验

（1）在`MemberServlet`中添加方法`checkLogin`方法，用于判断登录用户的id是否存在redis中

```
 public Result checkLogin(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        Member member = getData(request,Member.class);
        //根据获取到的id去redis中查找，是否存在
        String nickName = memberService.getLoginInfo(member.getId());
        return new Result("", nickName);
    }
```

（2）修改`index.html`中checkLogin()方法

```
 checkLogin() {
    //判断当前用户是否登录了，判断现在是否有登录人的信息

    /*  状态1：未登录
    document.querySelector("#register").style.display = 'block';
    document.querySelector("#login").style.display = 'block';
```

```
    document.querySelector("#myexam").style.display = 'none';
    document.querySelector("#exam").style.display = 'none';
    document.querySelector("#exit").style.display = 'none';
    document.querySelector("#nickname").style.display = 'none';
    */


    /*  状态2：已登录
    document.querySelector("#register").style.display = 'none';
    document.querySelector("#login").style.display = 'none';
    document.querySelector("#myexam").style.display = 'block';
    document.querySelector("#exam").style.display = 'block';
    document.querySelector("#exit").style.display = 'block';
    document.querySelector("#nickname").style.display = 'block';
    */

    let _this = this;
    //从localStorage中获取数据，获取当前保存的用户名，再根据用户名获取后台是否登录的状态
    if(!window.localStorage){
    alert("浏览器不支持localStorage，请升级浏览器")
    }else {
    //获取localStorage对象
    let storage = window.localStorage;
    alert("id: "+storage.id);

    //测试是否有登录数据，id
    if(storage.id ==  undefined){
    //如果本地没有用户信息，显示登陆和注册按钮
    document.querySelector("#register").style.display = 'block';
    document.querySelector("#login").style.display = 'block';
    document.querySelector("#myexam").style.display = 'none';
    document.querySelector("#exam").style.display = 'none';
    document.querySelector("#exit").style.display = 'none';
    document.querySelector("#nickname").style.display = 'none';
    }else {


    //如果本地存在用户信息，需要确认服务器是否存在当前用户登录信息（redis中）
    //发送请求，根据当前id去服务器中查找对应的数据
    axios.post('/member/checkLogin', '{"id":"'+storage.id+'"}').then(function
(response) {
    //alert(response.data.data)


    //获取响应数据
    let res = response.data;
    //alert("redis服务器中存储的用户名信息  : "+res.data)
    //判定本地用户对应是否处于登录状态，处于登录状态的用户具有用户名信息
    if(res.data == undefined){
    //如果没有用户名信息，当前用户未登录，显示登陆与注册按钮
    document.querySelector("#register").style.display = 'block';
    document.querySelector("#login").style.display = 'block';
    document.querySelector("#myexam").style.display = 'none';
    document.querySelector("#exam").style.display = 'none';
    document.querySelector("#exit").style.display = 'none';
    document.querySelector("#nickname").style.display = 'none';
    }else{
    //如果具有用户名信息，显示用户答题相关按钮
```

```
    //设置vue对象nickname属性值
    _this.nickname = res.data;
    document.querySelector("#register").style.display = 'none';
    document.querySelector("#login").style.display = 'none';
    document.querySelector("#myexam").style.display = 'block';
    document.querySelector("#exam").style.display = 'block';
    document.querySelector("#exit").style.display = 'block';
    //显示当前登录用户对应的登录信息组件
    document.querySelector("#nickname").style.display = 'block';
    }
    }).catch(function (err) {
    console.log(err)
    });
    }
    }
   },
```

（3）显示登录昵称

`MemberService`接口中添加方法`getLoginInfo`

```
/**
     * 根据登录人id获取对应的昵称，从redis中获取
     * @param id
     * @return
     */
    String getLoginInfo(String id);
```

修改`MemberServiceImpl`实现类中`login`方法

```
//使用登录人的id作为key，设定3600秒的过期时间，value值待定
jedis.setex(member.getId(),3600,member.getNickName());
```

添加方法`getLoginInfo`方法

```
 @Override
    public String getLoginInfo(String id) {
        //使用给定的id去查找redis中是否存在当前数据
        Jedis jedis = JedisUtils.getResource();
        String nickName = jedis.get(id);
        jedis.close();
        return nickName;
    }
```

1.4 退出登录

（1）修改`index.html`页面，将退出方法名修改成`logout`，并添加`logout`方法

```
<span class="top-right"  id="exit" style="display: none">
    <a href="#" @click="logout()" style="color:white">退出</a>
</span>
```

```
logout(){
    //1.获取localStorage
```

```
    let storage = window.localStorage;
    //2.发送请求，清除登录状态
    axios.post('/member/logout', '{"id":"'+storage.id+'"}').then(function
(response) {
        //1.获取响应数据
        let data = response.data;
        //2.提示
        //alert(data.flag);
    }).catch(function (err) {
        console.log(err)
    });
        //3.清理localStorage
        window.localStorage.clear();
        //4.通过调用checkLogin方法重置主页面的右上角显示区域
        this.checkLogin();
    }
},
```

(2) `MemberServlet` 添加 `logout` 方法

```
public Result logout(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        Member member = getData(request,Member.class);
        boolean flag = memberService.logout(member.getId());
        if(flag){
            return new Result("退出成功!", flag);
        }else{
            return new Result("", false, flag, Code.LOGOUT_FAIL);
        }
    }
```

`MemberService` 添加方法

```
boolean logout(String id);
```

`MemberServiceImpl` 添加方法

```
  @Override
    public boolean logout(String id) {
        Jedis jedis = JedisUtils.getResource();
        Long row = jedis.del(id);
        jedis.close();
        return row > 0 ;
    }
```

2.答题试卷

2.1 生成试卷

（1）创建实体类 `Question`

```
package com.itheima.domain.store;
import java.util.List;
public class Question {
    private String id;            //题目ID
    private String subject;     //题干
    private String type;        //题目类型  1:单选，2：多选，3：简答
}
```

(2) 创建`ExamServlet`继承`BaseServlet`

```
@WebServlet("/exam/*")
public class ExamServlet extends BaseServlet {
    public Result getPaper(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        List<Question> questionList = examService.getPaper();
        return new Result("试卷生成成功！", questionList);
    }
}
```

(3) 创建`ExamService`

```
public interface ExamService {
    List<Question> getPaper();
}
```

(4) 创建`ExamServiceImpl`实现`ExamService`

```
public class ExamServiceImpl implements ExamService {
    @Override
    public List<Question> getPaper() {
        SqlSession sqlSession = null;
        try{
            //1.获取SqlSession
            sqlSession = MapperFactory.getSqlSession();
            //2.获取Dao
            QuestionDao questionDao = MapperFactory.getMapper(sqlSession,
QuestionDao.class);
            List<Question> questionList = questionDao.findAll();
            return questionList;
        }catch (Exception e){
            e.printStackTrace();
            throw new RuntimeException(e);
            //记录日志
        }finally {
            try {
                TransactionUtil.close(sqlSession);
            }catch (Exception e){
                e.printStackTrace();
            }
        }
    }
}
```

(5) 创建`QuestionDao`

```
public interface QuestionDao {
    List<Question> findAll();
}
```

(6) 创建`QuestionDao.xml`

```
<?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.itheima.dao.store.QuestionDao">
    <!--配置实体类属性和数据库表中列的对应关系-->
    <resultMap id="BaseResultMap" type="com.itheima.domain.store.Question">
        <id column="id" jdbcType="VARCHAR" property="id"/>
        <result column="subject" jdbcType="VARCHAR" property="subject"/>
        <result column="type" jdbcType="VARCHAR" property="type"/>
    </resultMap>

    <!--配置查询的列名公共SQL语句-->
    <sql id="Base_Column_List">
        id, subject,type
    </sql>

    <!--配置查询所有，带条件-->
    <select id="findAll" resultMap="BaseResultMap">
        select
        <include refid="Base_Column_List"/>
        from st_question
        order by
        limit 2
    </select>
</mapper>
```

启动服务，开始测试

2.2 加载选项

（1）创建`QuestionItem`实体类

```
package com.itheima.domain.store;

public class QuestionItem {
    private String id;              //ID
    private String questionId;  //题目ID
    private String content;     //选项内容
    private String isRight;    //是否正确答案
}
```

（2）给`Question`实体类添加属性`questionItemList`

```
    private List<QuestionItem> questionItemList;

    public List<QuestionItem> getQuestionItemList() {
        return questionItemList;
    }

    public void setQuestionItemList(List<QuestionItem> questionItemList) {
        this.questionItemList = questionItemList;
    }
```

(3) 创建`QuestionItemDao`

```
public interface QuestionItemDao {
    List<QuestionItem> findByQuestionId(String questionId);
}
```

(4) 创建`QuestionItemDao.xml`

```
<?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.itheima.dao.store.QuestionItemDao">
    <!--配置实体类属性和数据库表中列的对应关系-->
    <resultMap id="BaseResultMap" type="com.itheima.domain.store.QuestionItem">
        <id column="id" jdbcType="VARCHAR" property="id"/>
        <result column="question_id" jdbcType="VARCHAR" property="questionId"/>
        <result column="content" jdbcType="VARCHAR" property="content"/>
        <result column="is_right" jdbcType="VARCHAR" property="isRight"/>
    </resultMap>

    <!--配置查询的列名公共SQL语句-->
    <sql id="Base_Column_List">
        id, question_id, content, is_right
    </sql>

    <!--配置查询所有，带条件-->
    <select id="findAll" parameterType="java.lang.String"
resultMap="BaseResultMap">
        select
        <include refid="Base_Column_List"/>
        from st_question_item
        where question_id = #{questionId,jdbcType=VARCHAR}
    </select>

    <!--配置根据ID查询-->
    <select id="findById" parameterType="java.lang.String"
resultMap="BaseResultMap">
        select
        <include refid="Base_Column_List"/>
        from st_question_item
        where id = #{id,jdbcType=VARCHAR}
    </select>

    <!--根据题目id查询所有选项-->
    <select id="findByQuestionId" resultMap="BaseResultMap"
parameterType="string">
```

```
        select
        <include refid="Base_Column_List"/>
        from st_question_item
        where question_id = #{questionId}
    </select>
</mapper>
```

(5) 修改`QuestionDao.xml`

```
<?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.itheima.dao.store.QuestionDao">
    <!--配置实体类属性和数据库表中列的对应关系-->
    <resultMap id="BaseResultMap" type="com.itheima.domain.store.Question">
        <id column="id" jdbcType="VARCHAR" property="id"/>
        <result column="subject" jdbcType="VARCHAR" property="subject"/>
        <result column="type" jdbcType="VARCHAR" property="type"/>

        <!--题目对题目选项的一对多-->
        <collection
            property="questionItemList"
            javaType="java.util.List"
            ofType="com.itheima.domain.store.QuestionItem"
            select="com.itheima.dao.store.QuestionItemDao.findByQuestionId"
            column="id"
            />
    </resultMap>

    <!--配置查询的列名公共SQL语句-->
    <sql id="Base_Column_List">
        id, subject,type
    </sql>

    <!--配置查询所有，带条件-->
    <select id="findAll" resultMap="BaseResultMap">
        select
        <include refid="Base_Column_List"/>
        from st_question
        order by id desc
        limit 2
    </select>
</mapper>
```

2.3 单选结果处理

在`paper.html`页面中，修改`changeResultRadio`方法

```
 changeResultRadio(item){
     //输出选中信息
     // alert("当前选项所属题目id：'+item.questionId);
     // alert('当前选项id：'+item.id);
     // alert("本次操作前数据结果：" + JSON.stringify(this.results))

    //组织数据（要加入到results中的数据）
    var temp = {"questionId":item.questionId,"answer":item.id};
```

```
    //alert(JSON.stringify(temp))

    //删除当前results中已经存在的本题目对应的数据
    this.results = this.results.filter(e=>{return e.questionId !==
item.questionId})
    //alert("results[操作前]：" + JSON.stringify(this.results))

    //将本次操作的结果加入到results
    this.results.push(temp);
    //alert("results[操作后]：" + JSON.stringify(this.results))
},
```

刷新页面，查看alert信息

2.4 多选结果处理

在 `paper.html` 页面中，修改 `changeResultCheckBox` 方法

```
            changeResultCheckBox(item){
                /*
                var s = '11,22,33,44';
                x = '33'
                var arr = s.split(",");
                var index = arr.indexOf(x);
                arr.splice(index,1);
                s = arr.join(",");
                alert(s)
                */


                //输出选中信息
                // alert("当前选项所属题目id：+item.questionId);
                // alert("当前选项id：+item.id);
                // alert('当前'+ this.checked);
                // alert("本次操作前数据结果：" + JSON.stringify(this.results))

                var temp = this.results.find(e=>{return e.questionId ===
item.questionId})

                if(temp == undefined){
                    //当前题目从来未作答过
                    temp = {"questionId":item.questionId,"answer":item.id};
                }else{
                    if(this.checked){
                        // 添加该答案
                        temp.answer = temp.answer + "," + item.id;
                    }else{
                        //删除该答案
                        var arr = temp.answer.split(",");
                        var index = arr.indexOf(item.id);
                        arr.splice(index,1);
                        temp.answer = arr.join(",");
                    }
                }

                //组织数据（要加入到results中的数据）
                // var temp = {"questionId":item.questionId,"answer":item.id};
```

```
                // alert(JSON.stringify(temp))

                //删除当前results中已经存在的本题目对应的数据
                this.results = this.results.filter(e=>{return e.questionId !==
item.questionId})
                alert("results[操作前]：" + JSON.stringify(this.results))

                //将本次操作的结果加入到results
                this.results.push(temp);
                alert("results[操作后]：" + JSON.stringify(this.results))

            },
```

2.5 提交试卷

`paper.html` 最终代码

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <!--<link rel="stylesheet" href="https://unpkg.com/element-ui/lib/theme-
chalk/index.css">-->
    <link rel="stylesheet" href="element-ui/lib/theme-chalk/index.css">
    <link rel="stylesheet" href="/css/tt.css">
    <link rel="stylesheet" type="text/css" href="./plugins/normalize-
css/normalize.css" />
    <link rel="stylesheet" type="text/css" href="./plugins/yui/cssgrids-min.css"
/>
    <link rel="stylesheet" type="text/css" href="./plugins/sui/sui.min.css" />
    <link rel="stylesheet" type="text/css" href="./plugins/sui/sui-
append.min.css" />
    <link rel="stylesheet" type="text/css" href="./plugins/font-
awesome/css/font-awesome.min.css" />
    <link rel="stylesheet" type="text/css" href="./css/widget-base.css" />
    <link rel="stylesheet" type="text/css" href="./css/widget-head-foot.css" />
    <link rel="stylesheet" type="text/css"
href="./plugins/bootstrap/dist/css/bootstrap.min.css" />
    <link rel="stylesheet" type="text/css" href="./css/page-sj-spit-index.css"
/>

    <title>自我测试</title>
    <script src="js/vue.js" type="text/javascript"></script>
    <script src="./js/axios-0.18.0.js" type="text/javascript"></script>
    <script src="element-ui/lib/index.js"></script>
</head>
<body>

<div id="app">
    <div style="height:100px;background-color:black;color:white;text-
align:center;font-size:35px">
        开    始    答    题
    </div>
```

```
    <el-form :model="ruleForm" :rules="rules" ref="ruleForm" label-width="100px"
class="demo-ruleForm">
        <div class="wrapper tag-item">
            <div class="fl left-list">
                <div class="tc-data-list">
                    <div class="tc-list">
                        <ul class="detail-list" v-for="question in questions">
                            <li class="qa-item">
                                <div class="fl record">
                                    <div class="number">
                                        <div class="border answer">
                                            <p class="usenum"></p>
                                            <p class="zannum"><strong> 题干
</strong> </p>
                                        </div>
                                        <hr/>
                                        <div class="border answer">
                                            <p class="zannum"><strong>选项
</strong> </p>
                                        </div>
                                    </div>
                                </div>
                                <div class="info">
                                    <p class="text">
                                        <b>{{question.subject}}</b>
                                    </p>
                                    <div class="other">
                                        <div class="fl date">
                                        <span>
                                            <el-form-item label="单选题"  v-
if="question.type == 1">
                                                <el-radio-group v-
model="question.id">
                                                    <el-radio v-for="
(item,index) in question.questionItemList"
                                                              :key="item.id"
 :label="item.content"
 :name="question.id"
                                                              v-model="checked"
 @change="changeResultRadio(item)">
                                                        {{item.content}}
                                                    </el-radio>
                                                </el-radio-group>
                                            </el-form-item>
                                            <el-form-item label="多选题" v-
if="question.type == 2" >
                                                <el-checkbox v-for="
(item,index) in question.questionItemList"
                                                             :key="item.id"
 :label="item.content"
 :name="question.id"
                                                             v-model="checked"
```

```
 @change="changeResultCheckBox(item)">
                                                        {{item.content}}
                                                    </el-checkbox>
                                                </el-form-item>
                                            </span>
                                            </div>
                                        </div>
                                    </div>
                                    <div class="clearfix"></div>
                                </li>
                            <ul class="detail-list" v-for="question in questions">
                            </ul>
                            <el-form-item>
                                <el-button type="primary"
@click="submitForm('ruleForm')">交卷</el-button>
                            </el-form-item>
                        </div>
                    </div>
                </div>
            </div>
        </el-form>
</div>

</body>
</html>
<script>
    /* 脚本中创建对象,处理业务 */
    new Vue({
        el: '#app',
        data: {
            questions:[],
            results:[],
            checked:[],
        },
        methods: {
            /*-------------------------单选多选题处理结构     开始-------------------
------
                //1.分情况处理，单选题与多选题处理方式不同
                //2.约定回传的答案格式         单选题由题号与选项组成，多选题由题号和多个选项
组成
                {
                    questionId:XXXXXXXX,
                    answer:mm
                }
                {
                    questionId:XXXXXXXX,
                    answer:mm,nn
                }
                //3.无论是单选还是多选，最终都是将所有题目的答案结果放置在一个数据中保存
（results），整体操作就是为results添加/修改数据
                [{},{},{}]
                //4.操作模式
                每次操作一个题目，先将当前题目对应的questionId在原始results中删除，添加新
数据到results中
                [1,1,1]
                单选：加入一个数据，如果之前有这个题目的数据，先删除再添加
                多选：加入一个数据，如果之前有这个题目的数据，先删除再添加
```

```
                ------------------------单选多选题处理结构     结束-----------------
--------*/

            changeResultRadio(item){
                //输出选中信息
                // alert("当前选项所属题目id：+item.questionId);
                // alert('当前选项id：'+item.id);
                // alert("本次操作前数据结果：" + JSON.stringify(this.results))

                //组织数据（要加入到results中的数据）
                var temp = {"questionId":item.questionId,"answer":item.id};
                //alert(JSON.stringify(temp))

                //删除当前results中已经存在的本题目对应的数据
                this.results = this.results.filter(e=>{return e.questionId !==
item.questionId})
                //alert("results[操作前]：" + JSON.stringify(this.results))

                //将本次操作的结果加入到results
                this.results.push(temp);
                //alert("results[操作后]：" + JSON.stringify(this.results))
            },

            changeResultCheckBox(item){
                /*
                var s = '11,22,33,44';
                x = '33'
                var arr = s.split(",");
                var index = arr.indexOf(x);
                arr.splice(index,1);
                s = arr.join(",");
                alert(s)
                */

                //输出选中信息
                // alert("当前选项所属题目id：+item.questionId);
                // alert("当前选项id：+item.id);
                // alert('当前'+ this.checked);
                // alert("本次操作前数据结果：" + JSON.stringify(this.results))

                var temp = this.results.find(e=>{return e.questionId ===
item.questionId})

                if(temp == undefined){
                    //当前题目从来未作答过
                    temp = {"questionId":item.questionId,"answer":item.id};
                }else{
                    if(this.checked){
                        // 添加该答案
                        temp.answer = temp.answer + "," + item.id;
                    }else{
                        //删除该答案
                        var arr = temp.answer.split(",");
                        var index = arr.indexOf(item.id);
                        arr.splice(index,1);
                        temp.answer = arr.join(",");
                    }
                }
```

```
                //组织数据（要加入到results中的数据）
                // var temp = {"questionId":item.questionId,"answer":item.id};
                // alert(JSON.stringify(temp))

                //删除当前results中已经存在的本题目对应的数据
                this.results = this.results.filter(e=>{return e.questionId !==
item.questionId})
                //alert("results[操作前]：" + JSON.stringify(this.results))

                //将本次操作的结果加入到results
                this.results.push(temp);
                //alert("results[操作后]：" + JSON.stringify(this.results))




                var arr ;
                var temp = this.results.find(e=>{return e.questionId ===
item.questionId})
                if(temp == undefined){
                    temp = {"questionId":item.questionId,"answer":item.id};
                }else{
                    if(this.checked){
                        temp.answer = temp.answer + "," + item.id;
                    }else{
                        arr = temp.answer.split(",");
                        var index = arr.indexOf(item.id);
                        arr.splice(index,1);
                        temp.answer = arr.join(",");
                    }
                }
                this.results = this.results.filter(e=>{return e.questionId !==
item.questionId})
                if(var.length > 0){
                    this.results.push(temp);
                }



                var temp = this.results.find(e=>{return e.questionId ===
item.questionId})
                if(temp == undefined){
                    temp = {"questionId":item.questionId,"answer":item.id};
                }else{
                    if(this.checked){
                        temp.answer = temp.answer + "," + item.id;
                    }else{
                        var arr = temp.answer.split(",");
                        var index = arr.indexOf(item.id);
                        arr.splice(index,1);
                        temp.answer = arr.join(",");
                    }
                }
                this.results = this.results.filter(e=>{return e.questionId !==
item.questionId})
                this.results.push(temp);
            },
```

```
            submitForm(formName) {
                if(this.results.length != this.questions.length){
                    alert("请检查题目是否全部选择");
                    return;
                }else{
                    if(window.confirm("确定交卷吗？")) {
                        //把json数据转成字符串
                        let str = JSON.stringify(this.results);
                        //获取localSotrage
                        let storage = window.localStorage;
                        //发送请求，交卷
                        axios.post('/exam/applyPaper',
'{"memberId":"'+storage.id+'","results":'+str+'}').then(function (response) {
                            //输出提示信息
                            //alert(response.data.message);
                            //发送完请求，跳转到交卷成功页面
                            window.open('/index.html', '_self');
                        }).catch(function (err) {
                            console.log(err)
                        });
                    }
                }
            },
            findQuestion(){
                let _this = this;
                //4.发送post请求，获取题目信息
                axios.post('/exam/getPaper').then(function (response) {
                    //5.得到响应数据
                    var res = response.data;
                    //alert(JSON.stringify(res));
                    _this.questions = res.data;
                }).catch(function (err) {
                    console.log(err)
                });
            }
        },
        created(){
            this.findQuestion();
        },
    });

</script>
```

`ExamServlet`

```
@WebServlet("/exam/*")
public class ExamServlet extends BaseServlet {

    public Result getPaper(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        List<Question> questionList = examService.getPaper();
        return new Result("试卷生成成功！", questionList);
    }
    public Result applyPaper(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        //memberId:?????,results:[{},{}]
        //1.得到全部请求的json数据
```

```
        String json = JSONObject.parseObject(request.getInputStream(),
String.class);
        //2.将json数据转换为json对象
        JSONObject jsonObject = JSON.parseObject(json);
        //3.获取当前提交试卷人的id
        String memberId = jsonObject.getObject("memberId", String.class);
        //4.获取当前提交的试卷信息
        JSONArray jsonArray = jsonObject.getJSONArray("results");
        List<ExamQuestion> examQuestionList =
jsonArray.toJavaList(ExamQuestion.class);

        boolean flag = examService.applyPaper(memberId,examQuestionList);

        return new Result("试卷提交成功！",flag);
    }
}
```

`ExamPaper`实体类

```
package com.itheima.domain.front;

import java.util.Date;

public class ExamPaper {
    private String id;
    private String memberId;
    private Date applyTime;
    private String state;   //1-可用  0-不可用
    private Integer score;
}
```

`ExamQuestion`实体类

```
package com.itheima.domain.front;

public class ExamQuestion {
    private String id;
    private String examPaperId;
    private String questionId;
    private String answer;
}
```

`ExamService`接口

```
package com.itheima.service.front;

import com.itheima.domain.front.ExamQuestion;
import com.itheima.domain.store.Question;

import java.util.List;

public interface ExamService {
```

```
    List<Question> getPaper();

    boolean applyPaper(String memberId, List<ExamQuestion> examQuestionList);
}
```

`ExamServiceImpl` 实现类

```
package com.itheima.service.front.impl;

import com.itheima.dao.front.ExamPaperDao;
import com.itheima.dao.front.ExamQuestionDao;
import com.itheima.dao.store.QuestionDao;
import com.itheima.domain.front.ExamPaper;
import com.itheima.domain.front.ExamQuestion;
import com.itheima.domain.store.Question;
import com.itheima.factory.MapperFactory;
import com.itheima.service.front.ExamService;
import com.itheima.utils.TransactionUtil;
import org.apache.ibatis.session.SqlSession;

import java.util.Date;
import java.util.List;
import java.util.UUID;

public class ExamServiceImpl implements ExamService {
    @Override
    public List<Question> getPaper() {
        SqlSession sqlSession = null;
        try{
            //1.获取SqlSession
            sqlSession = MapperFactory.getSqlSession();
            //2.获取Dao
            QuestionDao questionDao = MapperFactory.getMapper(sqlSession,
QuestionDao.class);
            List<Question> questionList = questionDao.findAll();
            return questionList;
        }catch (Exception e){
            e.printStackTrace();
            throw new RuntimeException(e);
            //记录日志
        }finally {
            try {
                TransactionUtil.close(sqlSession);
            }catch (Exception e){
                e.printStackTrace();
            }
        }
    }

    @Override
    public boolean applyPaper(String memberId, List<ExamQuestion>
examQuestionList) {
        SqlSession sqlSession = null;
        try{
            boolean flag = true;
            //1.获取SqlSession
```

```
            sqlSession = MapperFactory.getSqlSession();
            //2.获取Dao
            ExamPaperDao examPaperDao = MapperFactory.getMapper(sqlSession,
ExamPaperDao.class);
            ExamQuestionDao examQuestionDao =
MapperFactory.getMapper(sqlSession, ExamQuestionDao.class);
            //3.提交保存的试卷信息
            ExamPaper examPaper  = new ExamPaper();
            String paperId = UUID.randomUUID().toString();
            examPaper.setId(paperId);
            examPaper.setApplyTime(new Date());
            examPaper.setMemberId(memberId);
            examPaper.setState("1");
            flag = flag && examPaperDao.save(examPaper) > 0;
            //4.提交保存的试卷中的所有题目对应的答案信息
            for(ExamQuestion eq: examQuestionList) {
                eq.setId(UUID.randomUUID().toString());
                eq.setExamPaperId(paperId);
                flag = flag && examQuestionDao.save(eq) > 0;
            }
            TransactionUtil.commit(sqlSession);
            return flag;
        }catch (Exception e){
            TransactionUtil.rollback(sqlSession);
            throw new RuntimeException(e);
            //记录日志
        }finally {
            try {
                TransactionUtil.close(sqlSession);
            }catch (Exception e){
                e.printStackTrace();
            }
        }
    }
}
```

`ExamPaperDao` 接口

```
package com.itheima.dao.front;

import com.itheima.domain.front.ExamPaper;

public interface ExamPaperDao {
    int save(ExamPaper examPaper);
}
```

`ExamQuestionDao` 接口

```java
package com.itheima.dao.front;

import com.itheima.domain.front.ExamQuestion;

public interface ExamQuestionDao {
    int save(ExamQuestion eq);
}
```

`ExamPaperDao.xml` 文件

```xml
<?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.itheima.dao.front.ExamPaperDao">
    <insert id="save" parameterType="com.itheima.domain.front.ExamPaper">
        insert into tr_examination_paper (id, member_id, state, apply_time)
        values (#{id}, #{memberId}, #{state}, #{applyTime})
    </insert>
</mapper>
```

`ExamQuestionDao.xml` 文件

```xml
<?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.itheima.dao.front.ExamQuestionDao">
    <insert id="save" parameterType="com.itheima.domain.front.ExamQuestion">
        insert into tr_member_question (id, question_id,
examinationpaper_id,answer_result)
        values (#{id}, #{questionId}, #{examPaperId},#{answer})
    </insert>
</mapper>
```