

web综合案例

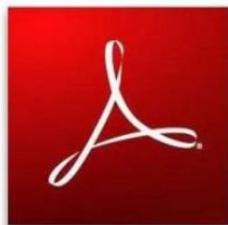
学习目标

- 目标1：完成使用POI读写Excel的测试案例
- 目标2：完成题目模板的制作，包括表头，标题及数据
- 目标3：完成题目报表数据导出的业务功能
- 目标4：完成角色与模块功能的快速开发
- 目标5：能够自己独立分析树形控件的页面制作
- 目标6：完成授权时动态加载授权数据
- 目标7：完成角色与模块的绑定关系

1. 报表

报表：简单的说，报表就是用表格、图表等格式来动态显示数据，可以用公式表示为：“报表 = 多样的格式 + 动态的数据”。

报表的种类有很多：Excel报表，PDF报表，网页报表等，他们各有优缺点



可编辑

动态数据

格式多样化

不可编辑

死数据

格式固定

不可编辑

动态数据

格式多样化

在本课程中，我们主要来将Excel报表。

对于Excel报表的技术实现上也有很多种选择：

- JXL：支持xls文件操作
- POI：支持xls和xlsx文件操作

我们只要来讲POI技术，要使用POI就要导入其坐标，如下

```
1 <!--POI-->
2 <dependency>
3     <groupId>org.apache.poi</groupId>
```

```

4     <artifactId>poi</artifactId>
5     <version>4.0.1</version>
6 </dependency>
7 <dependency>
8     <groupId>org.apache.poi</groupId>
9     <artifactId>poi-ooxml</artifactId>
10    <version>4.0.1</version>
11 </dependency>
12 <dependency>
13     <groupId>org.apache.poi</groupId>
14     <artifactId>poi-ooxml-schemas</artifactId>
15     <version>4.0.1</version>
16 </dependency>

```

1.1 POI写Excel文件

在测试包下创建POI测试类：com.itheima.service.store.PoiTest

```

1 public class PoiTest {
2
3     @Test
4     public void testWriteByPoi() throws IOException {
5         //1.获取到对应的Excel文件，工作簿文件
6         Workbook wb = new XSSFWorkbook();
7         //2.创建工作表
8         Sheet sheet = wb.createSheet();
9         wb.createSheet("这是啥呀");
10
11         //3.创建工作表中的行对象
12         Row row = sheet.createRow(1);
13         //4.创建工作表中行中的列对象
14         Cell cell = row.createCell(1);
15         //5.在列中写数据
16         cell.setCellValue("测试一下单元格");
17
18         //创建一个文件对象，作为excel文件内容的输出文件
19         File f = new File("test.xlsx");
20         //输出时通过流的形式对外输出，包装对应的目标文件
21         OutputStream os = new FileOutputStream(f);
22         //将内存中的workbook数据写入到流中
23         wb.write(os);
24         wb.close();
25         os.close();
26     }
27 }

```

使用单元测试进行测试！

1.2 POI读Excel文件

testReadByPoi

```
@Test
public void testReadByPoi() throws IOException {
    //1.获取要读取的文件工作簿对象
    Workbook wb = new XSSFWorkbook("test.xlsx");
    //2.获取工作表
    Sheet s = wb.getSheetAt(0);
    //3.获取行
    Row row = s.getRow(3);
    //4.获取列
    Cell cell = row.getCell(1);
    //5.根据数据的类型获取数据
    //      String data = cell.getStringCellValue();
    //      double data = cell.getNumericCellValue();
    boolean data = cell.getBooleanCellValue();

    System.out.println(data);

    wb.close();
}
```

直接读取第一节创建好的Excel文件

1.3 题目模板表头制作

前两节我们讲了如何去读取及写入Excel数据，操作相对简单，但是实际业务中我们要操作的Excel报表还是比较繁琐的，我们可以从今日课程资料中找到我们最终要导出报表的模板：`资料\Excel解析\模板.xlsx`

[illegible]

这种形式的我们如何去操作呢？

testProjectPoi

```
@Test
public void testProjectPoi() throws IOException {
    //1.获取到对应的Excel文件，工作簿文件
    Workbook wb = new XSSFWorkbook();
    //2.创建工作表
    Sheet s = wb.createSheet("题目数据文件");
    //制作标题
    s.addMergedRegion(new CellRangeAddress(1,1,1,12));
}
```

```

9      Row row_1 = s.createRow(1);
10     Cell cell_1_1 = row_1.createCell(1);
11     cell_1_1.setCellValue("在线试题导出信息");
12     //创建一个样式
13     CellStyle cs_title = wb.createCellStyle();
14     cs_title.setAlignment(HorizontalAlignment.CENTER);
15     cs_title.setVerticalAlignment(VerticalAlignment.CENTER);
16     cell_1_1.setCellStyle(cs_title);
17     //制作表头
18
19     //制作数据区
20
21     //创建一个文件对象，作为excel文件内容的输出文件
22     File f = new File("test.xlsx");
23     //输出时通过流的形式对外输出，包装对应的目标文件
24     OutputStream os = new FileOutputStream(f);
25     //将内存中的workbook数据写入到流中
26     wb.write(os);
27     wb.close();
28     os.close();
29 }

```

1.4 题目模板标题制作

下面我们接着来做Excel的表头

在测试方法 `testProjectPoi` 中继续编写代码

```

1  @Test
2  public void testProjectPoi() throws IOException {
3      //1.获取到对应的Excel文件，工作簿文件
4      Workbook wb = new XSSFWorkbook();
5      //2.创建工作表
6      Sheet s = wb.createSheet("题目数据文件");
7      //设置通用配置
8      //      s.setColumnWidth(4,100);
9      //制作标题
10     s.addMergedRegion(new CellRangeAddress(1,1,1,12));
11     Row row_1 = s.createRow(1);
12     Cell cell_1_1 = row_1.createCell(1);
13     cell_1_1.setCellValue("在线试题导出信息");
14     //创建一个样式
15     CellStyle cs_title = wb.createCellStyle();
16     cs_title.setAlignment(HorizontalAlignment.CENTER);
17     cs_title.setVerticalAlignment(VerticalAlignment.CENTER);
18     cell_1_1.setCellStyle(cs_title);
19     //制作表头
20     String[] fields = {"题目ID", "所属公司ID", "所属目录ID", "题目简介", "题干描述",
21                       "题干配图", "题目分析", "题目类型", "题目难度", "是否经典题", "题目状态", "审核状态"};
22     Row row_2 = s.createRow(2);
23     for (int i = 0; i < fields.length; i++) {
24         Cell cell_2_temp = row_2.createCell(1 + i); //++

```

```

25     cell_2_temp.setCellValue(fields[i]);    //++
26
27     CellStyle cs_field = wb.createCellStyle();
28     cs_field.setAlignment(HorizontalAlignment.CENTER);
29     cell_2_temp.setCellStyle(cs_field);
30 }
31
32 //制作数据区
33
34 //创建一个文件对象，作为excel文件内容的输出文件
35 File f = new File("test.xlsx");
36 //输出时通过流的形式对外输出，包装对应的目标文件
37 OutputStream os = new FileOutputStream(f);
38 //将内存中的workbook数据写入到流中
39 wb.write(os);
40 wb.close();
41 os.close();
42 }

```

1.5 题目模板数据制作

我们继续来做数据区

```

1  @Test
2  public void testProjectPoi() throws IOException {
3      //1.获取到对应的Excel文件，工作簿文件
4      Workbook wb = new XSSFWorkbook();
5      //2.创建工作表
6      Sheet s = wb.createSheet("题目数据文件");
7      //设置通用配置
8      //      s.setColumnWidth(4,100);
9      CellStyle cs_field = wb.createCellStyle();
10     cs_field.setAlignment(HorizontalAlignment.CENTER);
11     cs_field.setBorderTop(BorderStyle.THIN);
12     cs_field.setBorderBottom(BorderStyle.THIN);
13     cs_field.setBorderLeft(BorderStyle.THIN);
14     cs_field.setBorderRight(BorderStyle.THIN);
15
16
17     //制作标题
18     s.addMergedRegion(new CellRangeAddress(1,1,1,12));
19     Row row_1 = s.createRow(1);
20     Cell cell_1_1 = row_1.createCell(1);
21     cell_1_1.setCellValue("在线试题导出信息");
22     //创建一个样式
23     CellStyle cs_title = wb.createCellStyle();
24     cs_title.setAlignment(HorizontalAlignment.CENTER);
25     cs_title.setVerticalAlignment(VerticalAlignment.CENTER);
26     cell_1_1.setCellStyle(cs_title);
27
28     //制作表头

```

```

28     String[] fields = {"题目ID", "所属公司ID", "所属目录ID", "题目简介", "题干描述",
29         "题干配图", "题目分析", "题目类型", "题目难度", "是否经典题", "题目状态", "审核
    状态"};
30     Row row_2 = s.createRow(2);
31
32     for (int i = 0; i < fields.length; i++) {
33         Cell cell_2_temp = row_2.createCell(1 + i); //++
34         cell_2_temp.setCellValue(fields[i]); //++
35         cell_2_temp.setCellStyle(cs_field);
36     }
37
38
39     //制作数据区
40     List<Question> questionList = new ArrayList<>();
41     Question qq = new Question();
42     qq.setId("1");
43     qq.setPicture("12");
44     qq.setReviewStatus("13");
45     qq.setAnalysis("14");
46     qq.setCatalogId("15");
47     qq.setCompanyId("16");
48     qq.setDifficulty("17");
49     qq.setIsClassic("18");
50     qq.setRemark("19");
51     qq.setState("21");
52     qq.setSubject("31");
53     qq.setType("41");
54     questionList.add(qq);
55     Question qqq = new Question();
56     qqq.setId("1");
57     qqq.setPicture("12");
58     qqq.setReviewStatus("13");
59     qqq.setAnalysis("14");
60     qqq.setCatalogId("15");
61     qqq.setCompanyId("16");
62     qqq.setDifficulty("17");
63     qqq.setIsClassic("18");
64     qqq.setRemark("19");
65     qqq.setState("21");
66     qqq.setSubject("31");
67     qqq.setType("41");
68     questionList.add(qqq);
69
70
71     int row_index = 0;
72     for (Question q : questionList) {
73         int cell_index = 0;
74         Row row_temp = s.createRow(3 + row_index++);
75
76         Cell cell_data_1 = row_temp.createCell(1 + cell_index++);
77         cell_data_1.setCellValue(q.getId()); //++
78         cell_data_1.setCellStyle(cs_field);
79

```

```

80     Cell cell_data_2 = row_temp.createCell(1 + cell_index++);
81     cell_data_2.setCellValue(q.getCompanyId());    //++
82     cell_data_2.setCellStyle(cs_field);
83
84     Cell cell_data_3 = row_temp.createCell(1 + cell_index++);
85     cell_data_3.setCellValue(q.getCatalogId());    //++
86     cell_data_3.setCellStyle(cs_field);
87
88     Cell cell_data_4 = row_temp.createCell(1 + cell_index++);
89     cell_data_4.setCellValue(q.getRemark());    //++
90     cell_data_4.setCellStyle(cs_field);
91
92     Cell cell_data_5 = row_temp.createCell(1 + cell_index++);
93     cell_data_5.setCellValue(q.getSubject());    //++
94     cell_data_5.setCellStyle(cs_field);
95
96     Cell cell_data_6 = row_temp.createCell(1 + cell_index++);
97     cell_data_6.setCellValue(q.getPicture());    //++
98     cell_data_6.setCellStyle(cs_field);
99
100    Cell cell_data_7 = row_temp.createCell(1 + cell_index++);
101    cell_data_7.setCellValue(q.getAnalysis());    //++
102    cell_data_7.setCellStyle(cs_field);
103
104    Cell cell_data_8 = row_temp.createCell(1 + cell_index++);
105    cell_data_8.setCellValue(q.getType());    //++
106    cell_data_8.setCellStyle(cs_field);
107
108    Cell cell_data_9 = row_temp.createCell(1 + cell_index++);
109    cell_data_9.setCellValue(q.getDifficulty());    //++
110    cell_data_9.setCellStyle(cs_field);
111
112    Cell cell_data_10 = row_temp.createCell(1 + cell_index++);
113    cell_data_10.setCellValue(q.getIsClassic());    //++
114    cell_data_10.setCellStyle(cs_field);
115
116    Cell cell_data_11 = row_temp.createCell(1 + cell_index++);
117    cell_data_11.setCellValue(q.getState());    //++
118    cell_data_11.setCellStyle(cs_field);
119
120    Cell cell_data_12 = row_temp.createCell(1 + cell_index++);
121    cell_data_12.setCellValue(q.getReviewStatus());    //++
122    cell_data_12.setCellStyle(cs_field);
123 }
124
125 //创建一个文件对象，作为excel文件内容的输出文件
126 File f = new File("test.xlsx");
127 //输出时通过流的形式对外输出，包装对应的目标文件
128 OutputStream os = new FileOutputStream(f);
129 //将内存中的workbook数据写入到流中
130 wb.write(os);
131 wb.close();
132 os.close();

```

测试即可！

1.6 题目报表数据准备

(1) 找到 `/WEB-INF/pages/store/question/list.jsp` 页面，修改导出题目的链接

```
1 <button type="button" class="btn btn-default" title="导出题目"
  onclick=location.href="${ctx}/store/question?operation=downloadReport"> <i class="fa fa-
  download"></i>导出题目</button>
```

(2) 在后台servlet中添加对应的方法

```
1 // uri:/store/question?operation=list
2 @WebServlet("/store/question")
3 public class QuestionServlet extends BaseServlet {
4
5     @Override
6     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
  ServletException, IOException {
7         String operation = request.getParameter("operation");
8         if("list".equals(operation)){
9             this.list(request,response);
10        }
11        //其他的else if判断省略
12        else if("downloadReport".equals(operation)){
13            this.downloadReport(request,response);
14        }
15    }
16
17    private void downloadReport(HttpServletRequest request, HttpServletResponse response)
  throws IOException {
18        //生成报告的文件，然后传递到前端页面
19        questionService.getReport();
20    }
21 }
```

(3) 在业务层 `QuestionService` 添加一个方法 `getReport`

```
1 public void getReport() throws IOException;
```

(4) 在对应的实现类中去实现该方法，把之前在测试类中的测试方法 `testProjectPoi` 里面的所有代码拷贝过来，其中数据我们应该是从数据库中查询出来，因此调用dao完成数据的查询

```
1 @Override
2 public void getReport() throws IOException{
3     //获取对应要展示的数据
4     SqlSession sqlSession = null;
```



```

5         List<Question> questionList = null;
6         try{
7             //1.获取SqlSession
8             sqlSession = MapperFactory.getSqlSession();
9             //2.获取Dao
10            QuestionDao questionDao =
MapperFactory.getMapper(sqlSession,QuestionDao.class);
11            //3.调用Dao层操作
12            questionList = questionDao.findAll();
13        }catch (Exception e){
14            throw new RuntimeException(e);
15            //记录日志
16        }finally {
17            try {
18                TransactionUtil.close(sqlSession);
19            }catch (Exception e){
20                e.printStackTrace();
21            }
22        }
23
24
25        //1.获取到对应的Excel文件, 工作簿文件
26        Workbook wb = new XSSFWorkbook();
27        //2.创建工作表
28        Sheet s = wb.createSheet("题目数据文件");
29        //设置通用配置
30        // s.setColumnWidth(4,100);
31        CellStyle cs_field = wb.createCellStyle();
32        cs_field.setAlignment(HorizontalAlignment.CENTER);
33        cs_field.setBorderTop(BorderStyle.THIN);
34        cs_field.setBorderBottom(BorderStyle.THIN);
35        cs_field.setBorderLeft(BorderStyle.THIN);
36        cs_field.setBorderRight(BorderStyle.THIN);
37        //制作标题
38        s.addMergedRegion(new CellRangeAddress(1,1,1,12));
39        Row row_1 = s.createRow(1);
40        Cell cell_1_1 = row_1.createCell(1);
41        cell_1_1.setCellValue("在线试题导出信息");
42        //创建一个样式
43        CellStyle cs_title = wb.createCellStyle();
44        cs_title.setAlignment(HorizontalAlignment.CENTER);
45        cs_title.setVerticalAlignment(VerticalAlignment.CENTER);
46        cell_1_1.setCellStyle(cs_title);
47
48
49        //制作表头
50        String[] fields = {"题目ID","所属公司ID","所属目录ID","题目简介","题干描述",
51            "题干配图","题目分析","题目类型","题目难度","是否经典题","题目状态","审核状态"};
52        Row row_2 = s.createRow(2);
53        for (int i = 0; i < fields.length; i++) {
54            Cell cell_2_temp = row_2.createCell(1 + i); //++
55            cell_2_temp.setCellValue(fields[i]); //++
56
57            cell_2_temp.setCellStyle(cs_field);

```

```

57     }
58     //制作数据区
59     int row_index = 0;
60     for (Question q : questionList) {
61         int cell_index = 0;
62         Row row_temp = s.createRow(3 + row_index++);
63
64         Cell cell_data_1 = row_temp.createCell(1 + cell_index++);
65         cell_data_1.setCellValue(q.getId());    //++
66         cell_data_1.setCellStyle(cs_field);
67
68         Cell cell_data_2 = row_temp.createCell(1 + cell_index++);
69         cell_data_2.setCellValue(q.getCompanyId());    //++
70         cell_data_2.setCellStyle(cs_field);
71
72         Cell cell_data_3 = row_temp.createCell(1 + cell_index++);
73         cell_data_3.setCellValue(q.getCatalogId());    //++
74         cell_data_3.setCellStyle(cs_field);
75
76         Cell cell_data_4 = row_temp.createCell(1 + cell_index++);
77         cell_data_4.setCellValue(q.getRemark());    //++
78         cell_data_4.setCellStyle(cs_field);
79
80         Cell cell_data_5 = row_temp.createCell(1 + cell_index++);
81         cell_data_5.setCellValue(q.getSubject());    //++
82         cell_data_5.setCellStyle(cs_field);
83
84         Cell cell_data_6 = row_temp.createCell(1 + cell_index++);
85         cell_data_6.setCellValue(q.getPicture());    //++
86         cell_data_6.setCellStyle(cs_field);
87
88         Cell cell_data_7 = row_temp.createCell(1 + cell_index++);
89         cell_data_7.setCellValue(q.getAnalysis());    //++
90         cell_data_7.setCellStyle(cs_field);
91
92         Cell cell_data_8 = row_temp.createCell(1 + cell_index++);
93         cell_data_8.setCellValue(q.getType());    //++
94         cell_data_8.setCellStyle(cs_field);
95
96         Cell cell_data_9 = row_temp.createCell(1 + cell_index++);
97         cell_data_9.setCellValue(q.getDifficulty());    //++
98         cell_data_9.setCellStyle(cs_field);
99
100        Cell cell_data_10 = row_temp.createCell(1 + cell_index++);
101        cell_data_10.setCellValue(q.getIsClassic());    //++
102        cell_data_10.setCellStyle(cs_field);
103
104        Cell cell_data_11 = row_temp.createCell(1 + cell_index++);
105        cell_data_11.setCellValue(q.getState());    //++
106        cell_data_11.setCellStyle(cs_field);
107
108        Cell cell_data_12 = row_temp.createCell(1 + cell_index++);
109        cell_data_12.setCellValue(q.getReviewStatus());    //++

```

```

110         cell_data_12.setCellStyle(cs_field);
111     }
112
113     //创建一个文件对象，作为excel文件内容的输出文件
114     File f = new File("test.xlsx");
115     //输出时通过流的形式对外输出，包装对应的目标文件
116     OutputStream os = new FileOutputStream(f);
117     //将内存中的workbook数据写入到流中
118     wb.write(os);
119     wb.close();
120     os.close();
121 }

```

1.7 题目报表业务实现

现在后台已经能够生成Excel文件并且填充了数据，但是真实的业务中我们是需要将这个文件下载到客户端

(1) 修改接口方法 `getReport`，添加返回值

```

1  /**
2   * 获取包含了数据的流对象
3   * @return 包含了报表数据的流对象
4   * @throws IOException
5   */
6  ByteArrayOutputStream getReport() throws IOException;

```

(2) 在实现类中实现该方法时，将内存中的Excel相关数据写入到 `ByteArrayOutputStream` 流中

```

1  @Override
2  public ByteArrayOutputStream getReport() throws IOException {
3      //前面的代码无变动 故省略
4
5
6      /**
7      //创建一个文件对象，作为excel文件内容的输出文件
8          File f = new File("test.xlsx");
9          //输出时通过流的形式对外输出，包装对应的目标文件
10         OutputStream os = new FileOutputStream(f);
11         //将内存中的workbook数据写入到流中
12         wb.write(os);
13         wb.close();
14         os.close();
15
16         */
17         //将内存中的workbook数据写入到流中
18         ByteArrayOutputStream os = new ByteArrayOutputStream();
19         wb.write(os);
20         wb.close();
21         return os;
22     }

```

(3) 修改后台servlet的downloadReport方法

```
1 private void downloadReport(HttpServletRequest request, HttpServletResponse response) throws
  IOException {
2     //返回的数据类型为文件xlsx类型
3     response.setContentType("application/vnd.openxmlformats-
  officedocument.spreadsheetml.sheet;charset=utf-8");
4     String fileName = new String("测试文件名.xlsx".getBytes(),"iso8859-1");
5     response.addHeader("Content-Disposition","attachment;fileName="+fileName);
6
7     //生成报告的文件, 然后传递到前端页面
8     ByteArrayOutputStream os = questionService.getReport();
9     //获取产生响应的流对象
10    ServletOutputStream sos = response.getOutputStream();
11    //将数据从原始的字节流对象中提取出来写入到servlet对应的输出流中
12    os.writeTo(sos);
13    //将输出流刷新
14    sos.flush();
15    os.close();
16 }
```

(4) 启动项目, 进行测试

2.权限系统设计与开发

2.1 权限系统简介与结构设计

什么是权限系统?

权限系统是一种设定用户与可操作模块之间关系的系统。

通过设定用户与可操作的模块之间的关系,控制用户在可指定范围内进行业务执行

基于用户的权限控制(UBAC:User-BasedAccessControl)

基于角色的权限控制(RBAC:role-BasedAccessControl)

在本课程中我们采用基于角色的权限控制RBAC

用户表



用户-角色关系表

角色表



角色-模块关系表

模块表



2.2 角色与模块功能快速开发

首先来看角色与模块各自的结构

```
public class Role {  
    private String id;  
    private String name;  
    private String remark;  
    private Date createTime;  
}
```

名称
描述
创建时间

```
public class Module {  
    private String id;  
    private String parentId;  
    private String name;  
    private Long ctype;  
    private Long state;  
    private String curl;  
    private String remark;  
  
    private Module module;  
}
```

所属模块id
名称
类型（1-系统菜单，2-二级菜单，3-.....，4-.....）
状态（1-可用，2-不可用）
请求url（用于权限校验）
描述

自连接关系

(1) 创建角色实体：com.itheima.domain.system.Role

```
1 public class Role {  
2     private String id;  
3     private String name;  
4     private String remark;  
5     private Date createTime;  
6     // getter/setter略  
7 }
```

(2) 创建角色Dao：com.itheima.dao.system.RoleDao

```

1 public interface RoleDao {
2     int save(Role role);
3
4     int delete(Role role);
5
6     int update(Role role);
7
8     Role findById(String id);
9
10    List<Role> findAll();
11 }

```

(3) 添加接口的映射配置文件，从今日课程资料中找到 资料\dao层资源文件 将里面所有的xml映射配置文件拷贝到项目 `src/main/resources/com/itheima/dao/system` 目录下

(4) 创建业务层接口：com.itheima.service.system.RoleService

```

1 public interface RoleService {
2     /**
3      * 添加
4      * @param role
5      * @return
6      */
7     void save(Role role);
8
9     /**
10    * 删除
11    * @param role
12    * @return
13    */
14    void delete(Role role);
15
16    /**
17    * 修改
18    * @param role
19    * @return
20    */
21    void update(Role role);
22
23    /**
24    * 查询单个
25    * @param id 查询的条件 (id)
26    * @return 查询的结果, 单个对象
27    */
28    Role findById(String id);
29
30    /**
31    * 查询全部的数据
32    * @return 全部数据的列表对象
33    */
34    List<Role> findAll();
35 }

```

```

36  /**
37   * 分页查询数据
38   * @param page 页码
39   * @param size 每页显示的数据总量
40   * @return
41   */
42  PageInfo findAll(int page, int size);
43  }

```

(5) 创建接口的实现: com.itheima.service.system.impl

```

1  public class RoleServiceImpl implements RoleService {
2      @Override
3      public void save(Role role) {
4          SqlSession sqlSession = null;
5          try{
6              //1.获取SqlSession
7              sqlSession = MapperFactory.getSqlSession();
8              //2.获取Dao
9              RoleDao roleDao = MapperFactory.getMapper(sqlSession,RoleDao.class);
10             //id使用UUID的生成策略来获取
11             String id = UUID.randomUUID().toString();
12             role.setId(id);
13             //3.调用Dao层操作
14             roleDao.save(role);
15             //4.提交事务
16             TransactionUtil.commit(sqlSession);
17         }catch (Exception e){
18             TransactionUtil.rollback(sqlSession);
19             throw new RuntimeException(e);
20             //记录日志
21         }finally {
22             try {
23                 TransactionUtil.close(sqlSession);
24             }catch (Exception e){
25                 e.printStackTrace();
26             }
27         }
28     }
29
30     @Override
31     public void delete(Role role) {
32         SqlSession sqlSession = null;
33         try{
34             //1.获取SqlSession
35             sqlSession = MapperFactory.getSqlSession();
36             //2.获取Dao
37             RoleDao roleDao = MapperFactory.getMapper(sqlSession,RoleDao.class);
38             //3.调用Dao层操作
39             roleDao.delete(role);
40             //4.提交事务
41
42             TransactionUtil.commit(sqlSession);

```

```

42     }catch (Exception e){
43         TransactionUtil.rollback(sqlSession);
44         throw new RuntimeException(e);
45         //记录日志
46     }finally {
47         try {
48             TransactionUtil.close(sqlSession);
49         }catch (Exception e){
50             e.printStackTrace();
51         }
52     }
53 }
54
55 @Override
56 public void update(Role role) {
57     SqlSession sqlSession = null;
58     try{
59         //1.获取SqlSession
60         sqlSession = MapperFactory.getSqlSession();
61         //2.获取Dao
62         RoleDao roleDao = MapperFactory.getMapper(sqlSession,RoleDao.class);
63         //3.调用Dao层操作
64         roleDao.update(role);
65         //4.提交事务
66         TransactionUtil.commit(sqlSession);
67     }catch (Exception e){
68         TransactionUtil.rollback(sqlSession);
69         throw new RuntimeException(e);
70         //记录日志
71     }finally {
72         try {
73             TransactionUtil.close(sqlSession);
74         }catch (Exception e){
75             e.printStackTrace();
76         }
77     }
78 }
79
80 @Override
81 public Role findById(String id) {
82     SqlSession sqlSession = null;
83     try{
84         //1.获取SqlSession
85         sqlSession = MapperFactory.getSqlSession();
86         //2.获取Dao
87         RoleDao roleDao = MapperFactory.getMapper(sqlSession,RoleDao.class);
88         //3.调用Dao层操作
89         return roleDao.findById(id);
90     }catch (Exception e){
91         throw new RuntimeException(e);
92         //记录日志
93     }finally {
94         try {

```



```

95         TransactionUtil.close(sqlSession);
96     }catch (Exception e){
97         e.printStackTrace();
98     }
99 }
100 }
101
102 @Override
103 public List<Role> findAll() {
104     SqlSession sqlSession = null;
105     try{
106         //1.获取SqlSession
107         sqlSession = MapperFactory.getSqlSession();
108         //2.获取Dao
109         RoleDao roleDao = MapperFactory.getMapper(sqlSession,RoleDao.class);
110         //3.调用Dao层操作
111         return roleDao.findAll();
112     }catch (Exception e){
113         throw new RuntimeException(e);
114         //记录日志
115     }finally {
116         try {
117             TransactionUtil.close(sqlSession);
118         }catch (Exception e){
119             e.printStackTrace();
120         }
121     }
122 }
123
124 @Override
125 public PageInfo findAll(int page, int size) {
126     SqlSession sqlSession = null;
127     try{
128         //1.获取SqlSession
129         sqlSession = MapperFactory.getSqlSession();
130         //2.获取Dao
131         RoleDao roleDao = MapperFactory.getMapper(sqlSession,RoleDao.class);
132         //3.调用Dao层操作
133         PageHelper.startPage(page,size);
134         List<Role> all = roleDao.findAll();
135         PageInfo pageInfo = new PageInfo(all);
136         return pageInfo;
137     }catch (Exception e){
138         throw new RuntimeException(e);
139         //记录日志
140     }finally {
141         try {
142             TransactionUtil.close(sqlSession);
143         }catch (Exception e){
144             e.printStackTrace();
145         }
146     }
147 }

```

(6) 创建servlet: com.itheima.web.controller.system.RoleServlet

```

1  // uri:/system/role?operation=list
2  @WebServlet("/system/role")
3  public class RoleServlet extends BaseServlet {
4
5      @Override
6      protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
7          String operation = request.getParameter("operation");
8          if("list".equals(operation)){
9              this.list(request,response);
10         }else if("toAdd".equals(operation)){
11             this.toAdd(request,response);
12         }else if("save".equals(operation)){
13             this.save(request, response);
14         }else if("toEdit".equals(operation)){
15             this.toEdit(request,response);
16         }else if("edit".equals(operation)){
17             this.edit(request,response);
18         }else if("delete".equals(operation)){
19             this.delete(request,response);
20         }else if("author".equals(operation)){
21             this.author(request,response);
22         }
23     }
24
25     private void list(HttpServletRequest request,HttpServletResponse response) throws
ServletException, IOException {
26         //进入列表页
27         //获取数据
28         int page = 1;
29         int size = 5;
30         if(StringUtils.isNotBlank(request.getParameter("page"))){
31             page = Integer.parseInt(request.getParameter("page"));
32         }
33         if(StringUtils.isNotBlank(request.getParameter("size"))){
34             size = Integer.parseInt(request.getParameter("size"));
35         }
36         PageInfo all = roleService.findAll(page, size);
37         //将数据保存到指定的位置
38         request.setAttribute("page",all);
39         //跳转页面
40         request.getRequestDispatcher("/WEB-INF/pages/system/role/list.jsp").forward(request,response);
41     }
42
43     private void toAdd(HttpServletRequest request,HttpServletResponse response) throws
ServletException, IOException {
44         //加载所有的部门信息放入到roleList

```

```
45     List<Role> all = roleService.findAll();
46     request.setAttribute("roleList",all);
47     //跳转页面
48     request.getRequestDispatcher("/WEB-INF/pages/system/role/add.jsp").forward(request,response);
49 }
50
51 private void save(HttpServletRequest request,HttpServletResponse response) throws
ServletException, IOException {
52     //将数据获取到, 封装成一个对象
53     Role role = BeanUtil.fillBean(request,Role.class,"yyyy-MM-dd");
54     //调用业务层接口save
55     roleService.save(role);
56     //跳转回到页面list
57     response.sendRedirect(request.getContextPath()+"/system/role?operation=list");
58 }
59
60 private void toEdit(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
61     //查询要修改的数据findById
62     String id = request.getParameter("id");
63     Role role = roleService.findById(id);
64     //将数据加载到指定区域, 供页面获取
65     request.setAttribute("role",role);
66     //跳转页面
67     request.getRequestDispatcher("/WEB-INF/pages/system/role/update.jsp").forward(request,response);
68 }
69
70 private void edit(HttpServletRequest request, HttpServletResponse response) throws
IOException {
71     //将数据获取到, 封装成一个对象
72     Role role = BeanUtil.fillBean(request,Role.class,"yyyy-MM-dd");
73     //调用业务层接口save
74     roleService.update(role);
75     //跳转回到页面list
76     response.sendRedirect(request.getContextPath()+"/system/role?operation=list");
77 }
78
79 private void delete(HttpServletRequest request, HttpServletResponse response) throws
IOException {
80     //将数据获取到, 封装成一个对象
81     Role role = BeanUtil.fillBean(request,Role.class);
82     //调用业务层接口save
83     roleService.delete(role);
84     //跳转回到页面list
85     response.sendRedirect(request.getContextPath()+"/system/role?operation=list");
86 }
87
88 private void author(HttpServletRequest request, HttpServletResponse response) throws
IOException, ServletException {
89     //获取要授权的角色id
90
91     String roleId = request.getParameter("id");
```

```

91      //使用id查询对应的数据 (角色id对应的模块信息)
92      Role role = roleService.findById(roleId);
93      request.setAttribute("role",role);
94      //根据当前的角色id获取所有的模块数据, 并加载关系数据
95      List<Map> map = moduleService.findAuthorDataByRoleId(roleId);
96      //map转成json数据
97      ObjectMapper om = new ObjectMapper();
98      String json = om.writeValueAsString(map);
99      request.setAttribute("roleModuleJson",json);
100     // TODO 数据未查询
101     //跳转到树页面中
102     request.getRequestDispatcher("/WEB-INF/pages/system/role/author.jsp").forward(request,response);
103 }
104
105 @Override
106 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
107     this.doGet(request,response);
108 }
109 }

```

同时需要在BaseServlet中添加 RoleService

```

1  public class BaseServlet extends HttpServlet {
2      protected CompanyService companyService;
3      protected DeptService deptService;
4      protected UserService userService;
5      protected CourseService courseService;
6      protected CatalogService catalogService;
7      protected QuestionService questionService;
8      protected QuestionItemService questionItemService;
9      protected RoleService roleService;
10
11     @Override
12     public void init() throws ServletException {
13         companyService = new CompanyServiceImpl();
14         deptService = new DeptServiceImpl();
15         userService = new UserServiceImpl();
16         courseService = new CourseServiceImpl();
17         catalogService = new CatalogServiceImpl();
18         questionService = new QuestionServiceImpl();
19         questionItemService = new QuestionItemServiceImpl();
20         roleService = new RoleServiceImpl();
21     }
22 }

```

(7) 拷贝页面到项目中, 从今日课程资料中找到: 资料\模块页面 将下面所有模块全部拷贝到项目 /WEB-INF/pages/system 目录下

(8) 启动项目, 进行测试

然后我们按照相同的方式将模块的相关功能快速开发完成

(1) 创建模块实体: com.itheima.domain.system.Module

```
1 public class Module {
2     private String id;
3     private String parentId;
4     private String name;
5     private Long ctype;
6     private Long state;
7     private String curl;
8     private String remark;
9
10    private Module module;
11    // getter/setter略
12 }
```

(2) 创建模块dao: com.itheima.dao.system.ModuleDao

```
1 public interface ModuleDao {
2     int save(Module module);
3
4     int delete(Module module);
5
6     int update(Module module);
7
8     Module findById(String id);
9
10    List<Module> findAll();
11 }
```

(3) 映射配置文件, 之前已拷贝, 查看一下即可

(4) 创建业务层接口: com.itheima.service.system.ModuleService

```
1 public interface ModuleService {
2     /**
3      * 添加
4      * @param module
5      * @return
6      */
7     void save(Module module);
8
9     /**
10    * 删除
11    * @param module
12    * @return
13    */
14    void delete(Module module);
15 }
```

```

16  /**
17   * 修改
18   * @param module
19   * @return
20   */
21  void update(Module module);
22
23  /**
24   * 查询单个
25   * @param id 查询的条件 (id)
26   * @return 查询的结果, 单个对象
27   */
28  Module findById(String id);
29
30  /**
31   * 查询全部的数据
32   * @return 全部数据的列表对象
33   */
34  List<Module> findAll();
35
36  /**
37   * 分页查询数据
38   * @param page 页码
39   * @param size 每页显示的数据总量
40   * @return
41   */
42  PageInfo findAll(int page, int size);
43
44  }

```

(5) 创建业务层实现类: com.itheima.service.system.impl.ModuleServiceImpl

```

1  public class ModuleServiceImpl implements ModuleService {
2      @Override
3      public void save(Module module) {
4          SqlSession sqlSession = null;
5          try{
6              //1.获取SqlSession
7              sqlSession = MapperFactory.getSqlSession();
8              //2.获取Dao
9              ModuleDao moduleDao = MapperFactory.getMapper(sqlSession,ModuleDao.class);
10             //id使用UUID的生成策略来获取
11             String id = UUID.randomUUID().toString();
12             module.setId(id);
13             //3.调用Dao层操作
14             moduleDao.save(module);
15             //4.提交事务
16             TransactionUtil.commit(sqlSession);
17         }catch (Exception e){
18             TransactionUtil.rollback(sqlSession);
19             throw new RuntimeException(e);
20
21             //记录日志

```

```

21     }finally {
22         try {
23             TransactionUtil.close(sqlSession);
24         }catch (Exception e){
25             e.printStackTrace();
26         }
27     }
28 }
29
30 @Override
31 public void delete(Module module) {
32     SqlSession sqlSession = null;
33     try{
34         //1.获取SqlSession
35         sqlSession = MapperFactory.getSqlSession();
36         //2.获取Dao
37         ModuleDao moduleDao = MapperFactory.getMapper(sqlSession,ModuleDao.class);
38         //3.调用Dao层操作
39         moduleDao.delete(module);
40         //4.提交事务
41         TransactionUtil.commit(sqlSession);
42     }catch (Exception e){
43         TransactionUtil.rollback(sqlSession);
44         throw new RuntimeException(e);
45         //记录日志
46     }finally {
47         try {
48             TransactionUtil.close(sqlSession);
49         }catch (Exception e){
50             e.printStackTrace();
51         }
52     }
53 }
54
55 @Override
56 public void update(Module module) {
57     SqlSession sqlSession = null;
58     try{
59         //1.获取SqlSession
60         sqlSession = MapperFactory.getSqlSession();
61         //2.获取Dao
62         ModuleDao moduleDao = MapperFactory.getMapper(sqlSession,ModuleDao.class);
63         //3.调用Dao层操作
64         moduleDao.update(module);
65         //4.提交事务
66         TransactionUtil.commit(sqlSession);
67     }catch (Exception e){
68         TransactionUtil.rollback(sqlSession);
69         throw new RuntimeException(e);
70         //记录日志
71     }finally {
72         try {
73             TransactionUtil.close(sqlSession);

```

```

74         }catch (Exception e){
75             e.printStackTrace();
76         }
77     }
78 }
79
80 @Override
81 public Module findById(String id) {
82     SqlSession sqlSession = null;
83     try{
84         //1.获取SqlSession
85         sqlSession = MapperFactory.getSqlSession();
86         //2.获取Dao
87         ModuleDao moduleDao = MapperFactory.getMapper(sqlSession,ModuleDao.class);
88         //3.调用Dao层操作
89         return moduleDao.findById(id);
90     }catch (Exception e){
91         throw new RuntimeException(e);
92         //记录日志
93     }finally {
94         try {
95             TransactionUtil.close(sqlSession);
96         }catch (Exception e){
97             e.printStackTrace();
98         }
99     }
100 }
101
102 @Override
103 public List<Module> findAll() {
104     SqlSession sqlSession = null;
105     try{
106         //1.获取SqlSession
107         sqlSession = MapperFactory.getSqlSession();
108         //2.获取Dao
109         ModuleDao moduleDao = MapperFactory.getMapper(sqlSession,ModuleDao.class);
110         //3.调用Dao层操作
111         return moduleDao.findAll();
112     }catch (Exception e){
113         throw new RuntimeException(e);
114         //记录日志
115     }finally {
116         try {
117             TransactionUtil.close(sqlSession);
118         }catch (Exception e){
119             e.printStackTrace();
120         }
121     }
122 }
123
124 @Override
125 public PageInfo findAll(int page, int size) {
126     SqlSession sqlSession = null;

```



```

127     try{
128         //1.获取SqlSession
129         sqlSession = MapperFactory.getSqlSession();
130         //2.获取Dao
131         ModuleDao moduleDao = MapperFactory.getMapper(sqlSession,ModuleDao.class);
132         //3.调用Dao层操作
133         PageHelper.startPage(page,size);
134         List<Module> all = moduleDao.findAll();
135         PageInfo pageInfo = new PageInfo(all);
136         return pageInfo;
137     }catch (Exception e){
138         throw new RuntimeException(e);
139         //记录日志
140     }finally {
141         try {
142             TransactionUtil.close(sqlSession);
143         }catch (Exception e){
144             e.printStackTrace();
145         }
146     }
147 }
148 }

```

(6) 创建servlet: com.itheima.web.controller.system.ModuleServlet

```

1  // uri:/system/module?operation=list
2  @WebServlet("/system/module")
3  public class ModuleServlet extends BaseServlet {
4
5      @Override
6      protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
7          String operation = request.getParameter("operation");
8          if("list".equals(operation)){
9              this.list(request,response);
10         }else if("toAdd".equals(operation)){
11             this.toAdd(request,response);
12         }else if("save".equals(operation)){
13             this.save(request, response);
14         }else if("toEdit".equals(operation)){
15             this.toEdit(request,response);
16         }else if("edit".equals(operation)){
17             this.edit(request,response);
18         }else if("delete".equals(operation)){
19             this.delete(request,response);
20         }
21     }
22
23     private void list(HttpServletRequest request,HttpServletResponse response) throws
ServletException, IOException {
24         //进入列表页
25         //获取数据

```

```

26     int page = 1;
27     int size = 10;
28     if(StringUtils.isNotBlank(request.getParameter("page"))){
29         page = Integer.parseInt(request.getParameter("page"));
30     }
31     if(StringUtils.isNotBlank(request.getParameter("size"))){
32         size = Integer.parseInt(request.getParameter("size"));
33     }
34     PageInfo all = moduleService.findAll(page, size);
35     //将数据保存到指定的位置
36     request.setAttribute("page",all);
37     //跳转页面
38     request.getRequestDispatcher("/WEB-INF/pages/system/module/list.jsp").forward(request,response);
39 }
40
41 private void toAdd(HttpServletRequest request,HttpServletResponse response) throws
ServletException, IOException {
42     //加载所有的信息放入到moduleList
43     List<Module> all = moduleService.findAll();
44     request.setAttribute("moduleList",all);
45     //跳转页面
46     request.getRequestDispatcher("/WEB-INF/pages/system/module/add.jsp").forward(request,response);
47 }
48
49 private void save(HttpServletRequest request,HttpServletResponse response) throws
ServletException, IOException {
50     //将数据获取到, 封装成一个对象
51     Module module = BeanUtil.fillBean(request,Module.class,"yyyy-MM-dd");
52     //调用业务层接口save
53     moduleService.save(module);
54     //跳转回到页面list
55     response.sendRedirect(request.getContextPath()+"/system/module?operation=list");
56 }
57
58 private void toEdit(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
59     //查询要修改的数据findById
60     String id = request.getParameter("id");
61     Module module = moduleService.findById(id);
62     //将数据加载到指定区域, 供页面获取
63     request.setAttribute("module",module);
64     //跳转页面
65     request.getRequestDispatcher("/WEB-INF/pages/system/module/update.jsp").forward(request,response);
66 }
67
68 private void edit(HttpServletRequest request, HttpServletResponse response) throws
IOException {
69     //将数据获取到, 封装成一个对象
70     Module module = BeanUtil.fillBean(request,Module.class,"yyyy-MM-dd");
71     //调用业务层接口save

```

```

72     moduleService.update(module);
73     //跳转回到页面list
74     response.sendRedirect(request.getContextPath()+"/system/module?operation=list");
75 }
76
77     private void delete(HttpServletRequest request, HttpServletResponse response) throws
IOException {
78         //将数据获取到, 封装成一个对象
79         Module module = BeanUtil.fillBean(request,Module.class);
80         //调用业务层接口save
81         moduleService.delete(module);
82         //跳转回到页面list
83         response.sendRedirect(request.getContextPath()+"/system/module?operation=list");
84     }
85
86     @Override
87     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
88         this.doGet(request,response);
89     }
90 }

```

同时需要在BserServlet中添加 `ModuleService`

```

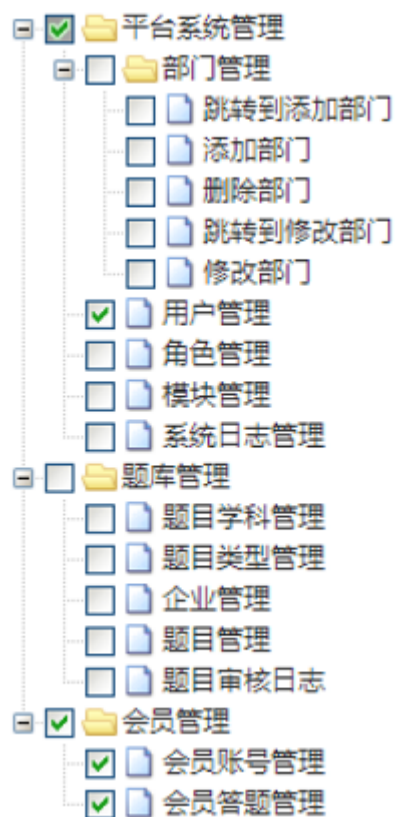
1  public class BaseServlet extends HttpServlet {
2      protected CompanyService companyService;
3      protected DeptService deptService;
4      protected UserService userService;
5      protected CourseService courseService;
6      protected CatalogService catalogService;
7      protected QuestionService questionService;
8      protected QuestionItemService questionItemService;
9      protected RoleService roleService;
10     protected ModuleService moduleService;
11
12     @Override
13     public void init() throws ServletException {
14         companyService = new CompanyServiceImpl();
15         deptService = new DeptServiceImpl();
16         userService = new UserServiceImpl();
17         courseService = new CourseServiceImpl();
18         catalogService = new CatalogServiceImpl();
19         questionService = new QuestionServiceImpl();
20         questionItemService = new QuestionItemServiceImpl();
21         roleService = new RoleServiceImpl();
22         moduleService = new ModuleServiceImpl();
23     }
24 }

```

(7) 拷贝页面：之前已经拷贝过了，我们可以直接启动项目进行测试！

2.3 树形控件结构分析（1）

树形结构如下图所示：



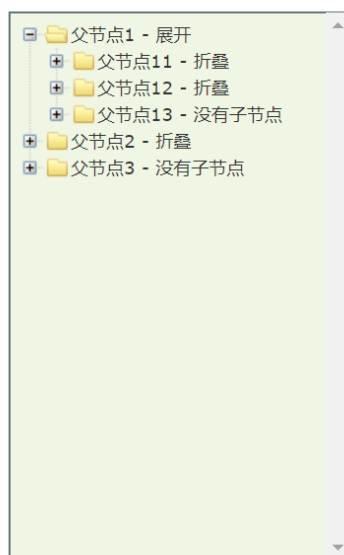
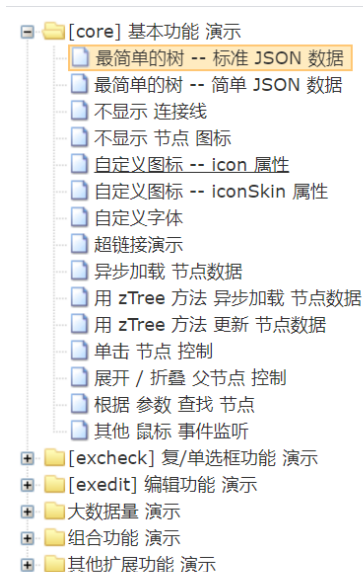
对应的实现技术有：

dTree

tdTree

zTree

我们主要来看关于**zTree**的相关操作，从今日课程资料中找到：[资料\树\zTree-zTree_v3-master\zTree_v3\demo\cn\index.html](#)，打开就可查阅



最简单的树 -- 标准 JSON 数据

[文件路径: core/standardData.html]

1、setting 配置信息说明

- 普通使用，无必须设置的参数
- 与显示相关的内容请参考 API 文档中 setting.view 内的配置信息
- name、children、title 等属性定义更改请参考 API 文档中 setting.data.key 内的配置信息

2、treeNode 节点数据说明

- 标准的 JSON 数据需要嵌套表示节点的父子包含关系
例如：

```
var nodes = [  
    {name: "父节点1", children: [  
        {name: "子节点1"},  
        {name: "子节点2"}  
    ]}  
];
```

- 默认展开的节点，请设置 treeNode.open 属性
- 无子节点的父节点，请设置 treeNode.isParent 属性
- 其他属性说明请参考 API 文档中 "treeNode 节点数据详解"

我们主要是针对里面的Checkbox 勾选操作进行学习，我们自己来编写一个测试页面 test.html 来完成一个树形结构，操作步骤：

- 1.观察整体的页面结构
- 2.去除无效的基础信息
- 3.去除页面无效的基础信息
- 4.分析页面js内容
- 5.分页结构所使用的数据
- 6.简化页面内容书写

```
1 - 1.观察整体的页面结构  
2 - 2.去除无效的基础信息  
3 - 3.去除页面无效的基础信息  
4 - 4.分析页面js内容  
5 - 5.分页结构所使用的数据  
6 - 6.简化页面内容书写  
7  
8 <meta http-equiv="content-type" content="text/html; charset=UTF-8">  
9 <link rel="stylesheet" href="../../css/demo.css" type="text/css">  
10 <link rel="stylesheet" href="../../css/zTreeStyle/zTreeStyle.css" type="text/css">  
11 <script type="text/javascript" src="../../js/jquery-1.4.4.min.js"></script>  
12 <script type="text/javascript" src="../../js/jquery.ztree.core-3.5.js"></script>  
13 <script type="text/javascript" src="../../js/jquery.ztree.excheck-3.5.js"></script>  
14 <SCRIPT type="text/javascript">  
15     var setting = {  
16         check: {  
17             enable: true  
18         },  
19         data: {  
20             simpleData: {  
21                 enable: true
```

```

22     }
23 }
24 };
25 /**/var zNodes =[
26     { id:11, pId:1, name:"随意勾选 1-1", open:true},
27     { id:111, pId:11, name:"随意勾选 1-1-1"},
28     { id:112, pId:11, name:"随意勾选 1-1-2"},
29     { id:12, pId:1, name:"随意勾选 1-2", open:true},
30     { id:121, pId:12, name:"随意勾选 1-2-1"},
31     { id:122, pId:12, name:"随意勾选 1-2-2"},
32     { id:2, pId:0, name:"随意勾选 2", checked:true, open:true},
33     { id:21, pId:2, name:"随意勾选 2-1"},
34     { id:22, pId:2, name:"随意勾选 2-2", open:true},
35     { id:221, pId:22, name:"随意勾选 2-2-1", checked:true},
36     { id:222, pId:22, name:"随意勾选 2-2-2"},
37     { id:23, pId:2, name:"随意勾选 2-3"},
38     { id:1, pId:0, name:"随意勾选 1", open:true}
39 ];
40 var code;
41 function setCheck() {
42     var zTree = $.fn.zTree.getZTreeObj("treeDemo"),
43     py = $("#py").attr("checked")? "p":"" ,
44     sy = $("#sy").attr("checked")? "s":"" ,
45     pn = $("#pn").attr("checked")? "p":"" ,
46     sn = $("#sn").attr("checked")? "s":"" ,
47     type = { "Y":py + sy, "N":pn + sn};
48     zTree.setting.check.chkboxType = type;
49     showCode('setting.check.chkboxType = { "Y" : "' + type.Y + '", "N" : "' + type.N +
50     "' }');
51 }
52 function showCode(str) {
53     if (!code) code = $("#code");
54     code.empty();
55     code.append("<li>"+str+"</li>");
56 }
57 $(document).ready(function(){
58     $.fn.zTree.init($("#treeDemo"), setting, zNodes);
59     setCheck();
60     $("#py").bind("change", setCheck);
61     $("#sy").bind("change", setCheck);
62     $("#pn").bind("change", setCheck);
63     $("#sn").bind("change", setCheck);
64 });
65 </SCRIPT>
66 <div class="content_wrap">
67     <div class="zTreeDemoBackground left">
68         <ul id="treeDemo" class="ztree"></ul>
69     </div>
70     <div class="right">
71         <ul class="info">
72             <li class="title">
73                 <ul class="list">

```

```

74         <input type="checkbox" id="py" class="checkbox first" checked />
<span>关联父</span>
75         <input type="checkbox" id="sy" class="checkbox first" checked />
<span>关联子</span><br/>
76         <input type="checkbox" id="pn" class="checkbox first" checked />
<span>关联父</span>
77         <input type="checkbox" id="sn" class="checkbox first" checked />
<span>关联子</span><br/>
78         <ul id="code" class="log" style="height:20px;"></ul></p>
79     </li>
80 </ul>
81 </li>
82 </ul>
83 </div>
84 </div>
85

```

2.4 树形控件结构分析（2）

分析页面js

```

1  - 1.观察整体的页面结构
2  - 2.去除无效的基础信息
3  - 3.去除页面无效的基础信息
4  - 4.分析页面js内容
5  - 5.分页结构所使用的数据
6  - 6.简化页面内容书写
7  <meta http-equiv="content-type" content="text/html; charset=UTF-8">
8  <link rel="stylesheet" href="../../../css/demo.css" type="text/css">
9  <link rel="stylesheet" href="../../../css/zTreeStyle/zTreeStyle.css" type="text/css">
10 <script type="text/javascript" src="../../../js/jquery-1.4.4.min.js"></script>
11 <script type="text/javascript" src="../../../js/jquery.ztree.core-3.5.js"></script>
12 <script type="text/javascript" src="../../../js/jquery.ztree.excheck-3.5.js"></script>
13 <SCRIPT type="text/javascript">
14     var setting = {check: {enable: true},data: {    simpleData: {enable: true}}};
15     var zNodes =[
16         { id:11, pId:1, name:"随意勾选 1-1", open:true},
17         { id:111, pId:11, name:"随意勾选 1-1-1"},
18         { id:112, pId:11, name:"随意勾选 1-1-2"},
19         { id:12, pId:1, name:"随意勾选 1-2", open:true},
20         { id:121, pId:12, name:"随意勾选 1-2-1"},
21         { id:122, pId:12, name:"随意勾选 1-2-2"},
22         { id:2, pId:0, name:"随意勾选 2", checked:true, open:true},
23         { id:21, pId:2, name:"随意勾选 2-1"},
24         { id:22, pId:2, name:"随意勾选 2-2", open:true},
25         { id:221, pId:22, name:"随意勾选 2-2-1", checked:true},
26         { id:222, pId:22, name:"随意勾选 2-2-2"},
27         { id:23, pId:2, name:"随意勾选 2-3"},
28         { id:1, pId:0, name:"随意勾选 1", open:true}
29     ];
30
31     $(document).ready(function(){

```

```

32     $.fn.zTree.init($("#treeDemo"), setting, zNodes);
33     var zTree = $.fn.zTree.getZTreeObj("treeDemo")
34     zTree.setting.check.chkboxType = { "Y" : "ps", "N" : "ps" }
35     });
36 </SCRIPT>
37 <ul id="treeDemo" class="ztree"></ul>

```

2.5 树形控件结构分析 (3)

继续进行数据结构的分析

```

1  <meta http-equiv="content-type" content="text/html; charset=UTF-8">
2  <link rel="stylesheet" href="../../css/demo.css" type="text/css">
3  <link rel="stylesheet" href="../../css/zTreeStyle/zTreeStyle.css" type="text/css">
4  <script type="text/javascript" src="../../js/jquery-1.4.4.min.js"></script>
5  <script type="text/javascript" src="../../js/jquery.ztree.core-3.5.js"></script>
6  <script type="text/javascript" src="../../js/jquery.ztree.excheck-3.5.js"></script>
7  <SCRIPT type="text/javascript">
8      var setting = {check: {enable: true},data: {    simpleData: {enable: true}}};
9      var zNodes =[
10         { id:2, pId:0, name:"test", checked:true, open:true},
11         { id:21, pId:2, name:"test2222"},
12         { id:22, pId:1, name:"test2222"}
13     ];
14
15     $(document).ready(function(){
16         $.fn.zTree.init($("#treeDemo"), setting, zNodes);
17         var zTree = $.fn.zTree.getZTreeObj("treeDemo")
18         zTree.setting.check.chkboxType = { "Y" : "ps", "N" : "ps" }
19     });
20 </SCRIPT>
21 <ul id="treeDemo" class="ztree"></ul>

```

2.6 动态加载授权数据

(1) 查看页面: `/WEB-INF/pages/system/role/list.jsp`, 授权按钮点击时要传递id

(2) 进入后台servlet: `RoleServlet` 添加 `author` 方法

```

1  @Override
2  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
3      String operation = request.getParameter("operation");
4      if("list".equals(operation)){
5          this.list(request,response);
6      }else if("toAdd".equals(operation)){
7          this.toAdd(request,response);
8      }else if("save".equals(operation)){
9          this.save(request, response);
10     }else if("toEdit".equals(operation)){
11         this.toEdit(request,response);

```



```

12     }else if("edit".equals(operation)){
13         this.edit(request,response);
14     }else if("delete".equals(operation)){
15         this.delete(request,response);
16     }else if("author".equals(operation)){
17         this.author(request,response);
18     }
19 }
20
21 private void author(HttpServletRequest request, HttpServletResponse response) throws
IOException, ServletException {
22     //获取要授权的角色id
23     String roleId = request.getParameter("id");
24
25     // TODO 数据未查询
26     //跳转到树页面中
27     request.getRequestDispatcher("/WEB-
INF/pages/system/role/author.jsp").forward(request,response);
28 }

```

(3) 在 `/WEB-INF/pages/system/role` 下创建一个jsp页面: `test.jsp`, 内容粘贴我们之前编辑的 `test.html` 页面, 我们在后台跳转的时候跳转的是该目录下的 `author.jsp`, 我们可以拿这两页面做一个对比

(4) 完善servlet中的 `author` 方法

```

1 private void author(HttpServletRequest request, HttpServletResponse response) throws
IOException, ServletException {
2     //获取要授权的角色id
3     String roleId = request.getParameter("id");
4     //使用id查询对应的数据 (角色id对应的模块信息)
5     Role role = roleService.findById(roleId);
6     request.setAttribute("role",role);
7     //根据当前的角色id获取所有的模块数据, 并加载关系数据
8     List<Map> map = moduleService.findAuthorDataByRoleId(roleId);
9     //map转成json数据
10    ObjectMapper om = new ObjectMapper();
11    String json = om.writeValueAsString(map);
12    request.setAttribute("roleModuleJson",json);
13    // TODO 数据未查询
14    //跳转到树页面中
15    request.getRequestDispatcher("/WEB-
INF/pages/system/role/author.jsp").forward(request,response);
16 }

```

在 `WEB-INF\pages\system\role\author.jsp` 页面中修改js代码: 用后台查询的数据直接赋值给 `zNodes`

```

1 var zNodes = ${roleModuleJson}

```

(5) 在 `ModuleService` 中添加 `findAuthorDataByRoleId` 方法

```

1  /**
2   * 根据角色id获取对应的所有模块关联数据
3   * @param roleId 角色id
4   */
5  List<Map> findAuthorDataByRoleId(String roleId);

```

(6) 在实现类中实现该方法

```

1  @Override
2  public List<Map> findAuthorDataByRoleId(String roleId) {
3      SqlSession sqlSession = null;
4      try{
5          //1.获取SqlSession
6          sqlSession = MapperFactory.getSqlSession();
7          //2.获取Dao
8          ModuleDao moduleDao = MapperFactory.getMapper(sqlSession,ModuleDao.class);
9          //3.调用Dao层操作
10         return moduleDao.findAuthorDataByRoleId(roleId);
11     }catch (Exception e){
12         throw new RuntimeException(e);
13         //记录日志
14     }finally {
15         try {
16             TransactionUtil.close(sqlSession);
17         }catch (Exception e){
18             e.printStackTrace();
19         }
20     }
21 }

```

(7) 添加dao接口方法: `findAuthorDataByRoleId`

```

1  List<Map> findAuthorDataByRoleId(String roleId);

```

(8) 在ModuleDao对应的映射配置文件中添加对应的查询语句

```

1  <select id="findAuthorDataByRoleId" parameterType="string" resultType="java.util.Map">
2      select
3          module_id as id,
4          parent_id as pId,
5          name as name,
6          case
7              when module_id in (select module_id from ss_role_module where role_id = #
{roleId})
8                  then 'true'
9                  else 'false'
10             end
11          as checked
12      from
13          ss_module

```

```
14 </select>
```

(9) 启动测试

2.7 绑定角色与模块关系

(1) 查看 WEB-INF\pages\system\role\author.jsp 页面中提交保存的js代码

```
1 <SCRIPT type="text/javascript">
2     //实现权限分配
3     function submitCheckedNodes() {
4         //1.获取所有的勾选权限节点
5         var nodes = zTreeObj.getCheckedNodes(true); //true:被勾选, false: 未被勾选
6         //2.循环nodes, 获取每个节点的id, 并将数据加入数组
7         //1,2,3,4,5      1+,"+2+","+3+....
8         //数据的临时存储数组, 为了方便内容连接成为一个由逗号分隔的字符串
9         var moduleArrays = [];
10        for(var i=0;i<nodes.length;i++) {
11            moduleArrays.push(nodes[i].id);
12        }
13        //3.将数组中的数据使用,连接后, 赋值给表单, 传入后台
14        $("#moduleIds").val(moduleArrays.join(',')); //1,2,3,4,5
15        $("#icform").submit();
16    }
17 </SCRIPT>
18 <form id="icform" method="post" action="${ctx}/system/role?operation=updateRoleModule">
19     <input type="hidden" name="roleId" value="${role.id}"/>
20     <input type="hidden" id="moduleIds" name="moduleIds" value=""/>
21     <ul id="treeDemo" class="ztree"></ul>
22 </form>
23 <!-- 工具栏 -->
24 </form>
```

(2) 在后台servlet中添加方法 updateRoleModule

```
1 @Override
2 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
3     String operation = request.getParameter("operation");
4     if("list".equals(operation)){
5         this.list(request,response);
6     }
7     //中间的else if无变动 省略
8     else if("updateRoleModule".equals(operation)){
9         this.updateRoleModule(request,response);
10    }
11 }
12 private void updateRoleModule(HttpServletRequest request, HttpServletResponse response)
throws IOException, ServletException {
13     String roleId = request.getParameter("roleId");
14     String moduleIds = request.getParameter("moduleIds");
```

```

15     roleService.updateRoleModule(roleId,moduleIds);
16     //跳转回到页面list
17     response.sendRedirect(request.getContextPath()+"/system/role?operation=list");
18 }

```

(3) 在 `RoleService` 中添加方法 `updateRoleModule`

```

1  /**
2   * 建立角色与模块之间的关联
3   * @param roleId 角色id
4   * @param moduleIds 模块id (多个)
5   */
6  void updateRoleModule(String roleId, String moduleIds);

```

(4) 在对应的实现类中实现该方法

```

1  @Override
2  public void updateRoleModule(String roleId, String moduleIds) {
3      SqlSession sqlSession = null;
4      try{
5          //1.获取SqlSession
6          sqlSession = MapperFactory.getSqlSession();
7          //2.获取Dao
8          RoleDao roleDao = MapperFactory.getMapper(sqlSession,RoleDao.class);
9          //3.调用Dao层操作
10         //修改role_module
11         //3.1现有的关系全部取消掉
12         roleDao.deleteRoleModule(roleId);
13         //3.2建立新的关系 (多个)
14         String[] moduleArray = moduleIds.split(",");
15         for(String moduleId:moduleArray){
16             roleDao.saveRoleModule(roleId,moduleId);
17         }
18         //4.提交事务
19         TransactionUtil.commit(sqlSession);
20     }catch (Exception e){
21         TransactionUtil.rollback(sqlSession);
22         throw new RuntimeException(e);
23         //记录日志
24     }finally {
25         try {
26             TransactionUtil.close(sqlSession);
27         }catch (Exception e){
28             e.printStackTrace();
29         }
30     }
31 }

```

(5) 在 `RoleDao` 中添加方法 `deleteRoleModule` , `saveRoleModule`

```
1 void deleteRoleModule(String roleId);
2
3 void saveRoleModule(@Param("roleId") String roleId, @Param("moduleId") String moduleId);
```

(6) 在对应的映射配置文件中添加对应的操作

```
1 <!--配置根据roleId删除关系表数据-->
2 <delete id="deleteRoleModule" parameterType="java.lang.String">
3     delete from ss_role_module
4     where role_id = #{roleId,jdbcType=VARCHAR}
5 </delete>
6
7 <!--配置全字段插入, 当某个字段没有值时, 插入null-->
8 <insert id="saveRoleModule" parameterType="map">
9     insert into ss_role_module (role_id, module_id)
10    values (#{roleId,jdbcType=VARCHAR}, #{moduleId,jdbcType=VARCHAR})
11 </insert>
```

(7) 启动项目进行测试