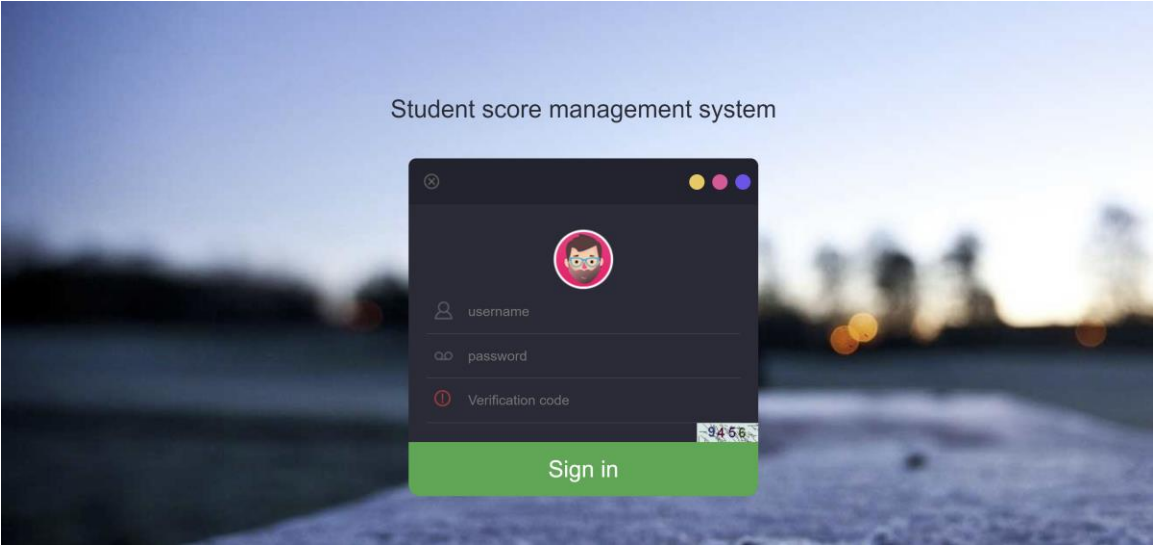


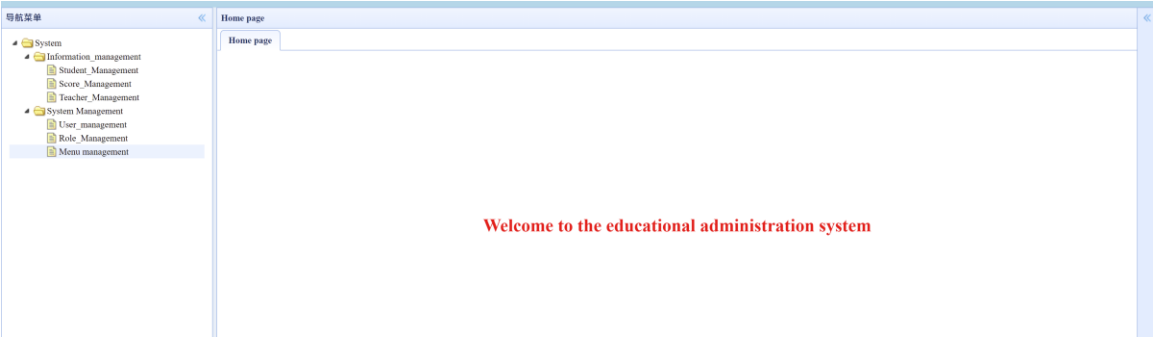
educational administration system

- **Short summary of the project**
Nowadays, the number of students on university campuses is increasing, and there are more and more types of disciplines. Inquiry and management of student performance has become more cumbersome, wastes a lot of human and material resources, and is prone to errors. Therefore, the development of a software for student performance Data management facilitates the extraction and operation of various information, so it is imperative to build a student management system. The development of this system can reduce the work pressure of faculty and staff, and systematically manage various services and information in educational administration and teaching, thereby improving confidentiality, speeding up inquiries, and improving management efficiency.
- **Summary of the Functionality Performed**
This system uses MySQL as the back-end database, easyui as the front-end technology, and eclipse as the development tool. The system realizes the necessary functions such as teacher management, student management, class management, course management, and student performance reports, export/import file in excel format. In addition, It can assign different levels of authority to different roles in order to make the system work in a more efficient and effective way.
- **Technologies used**
Java, MySQL database, easy-ui , JQuery, css, Spring MVC, Hibernate
- **User Roles and performed tasks for each role**
Admin:
 1. Basic information management(search/add/update/delete) including Score\Teacher\Student Info management
 2. System management: Assign different levels of authority
 3. Teacher: Limited information management(search/add/update/delete) including Score\Student Info management
- **Screenshots of key screens if available**

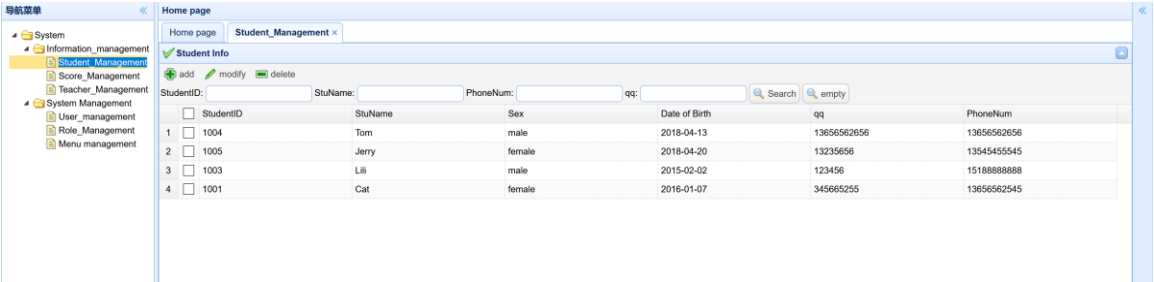
Login page



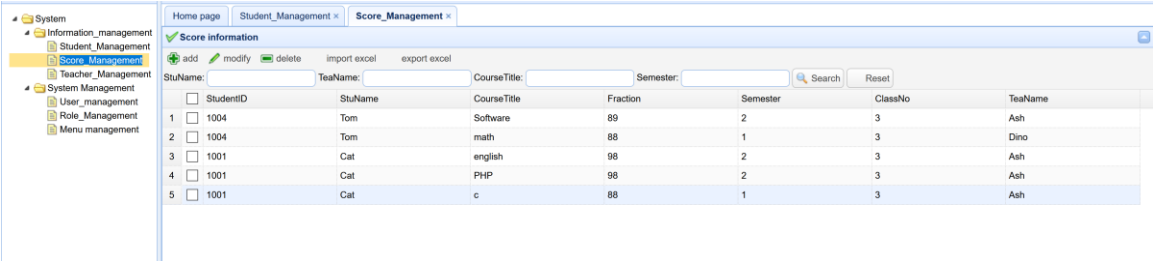
Admin role’s homepage



Student management



Score management



Teacher management

导航菜单

System

Information_management

Student_Management

Score_Management

Teacher_Management

System_Management

User_management

Role_Management

Menu_management

Home page

Student_Management

Score_Management

Teacher_Management

Teacher Info

add

modify

delete

TeacherID:

TeaName:

Search

reset

	TeacherID	TeaName	PhoneNum	Job Title	Entry Time	Creation time
1	<input type="checkbox"/> 103	Robin	18888888888	Senior	2018-04-01	2018-04-13 15:07:50
2	<input type="checkbox"/> 102	Ash	15166666666	Professor	2014-01-16	2016-02-01 10:47:30
3	<input type="checkbox"/> 101	Dino	13588888888	Professor	2009-11-18	2016-02-01 10:33:43

add

add

StudentID:

StuName:

Sex:

Date of Birth:

qq:

PhoneNum:

save

cancel

edit

edit

StudentID:

1004

StuName:

Tom


sex:

male

▼

Date of birth:

2018-04-13





qq:

13656562656

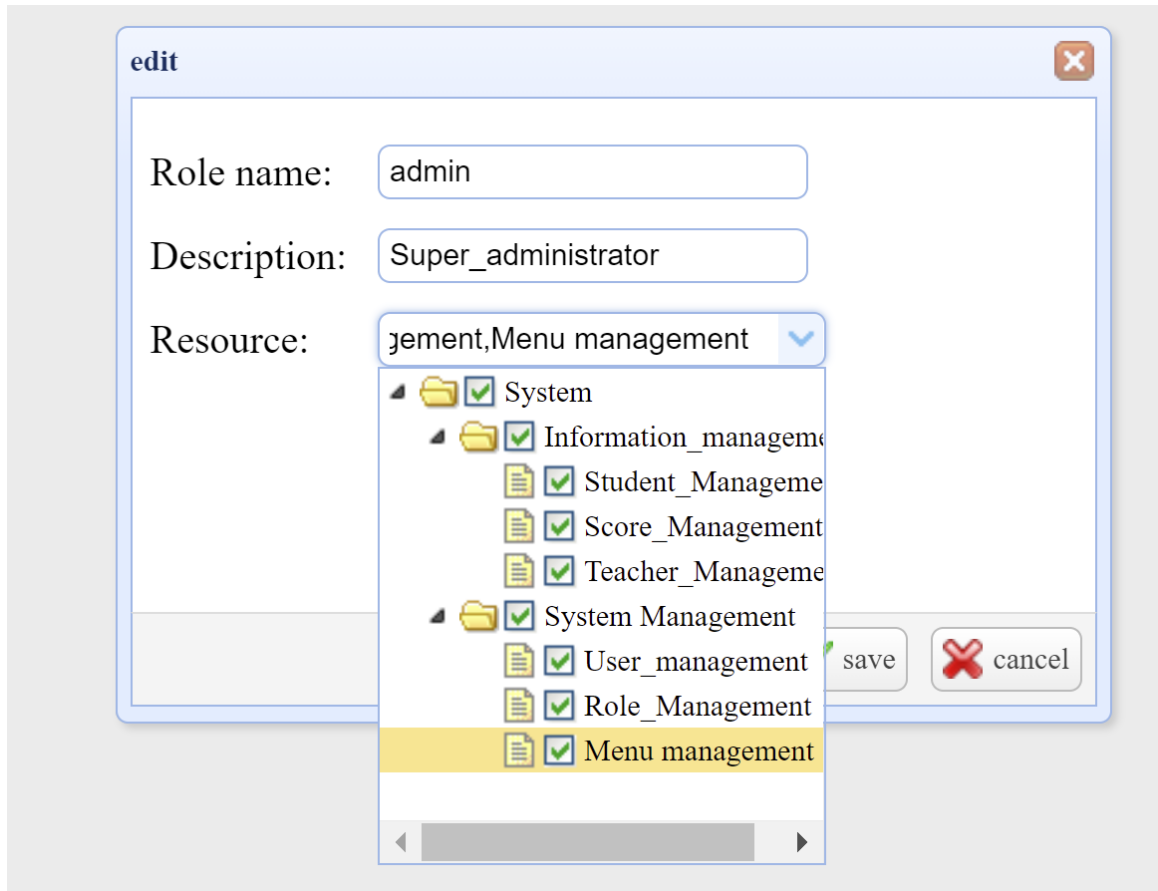
PhoneNum:

13656562656

 save

 cancel

page assigning different levels of authority



Menu management—set the menu's layout

导航菜单

System

Information_management

Student_Management

Score_Management

Teacher_Management

System Management

User_management

Role_Management

Menu management

Home page

Home page

Student_Management

Score_Management

Teacher_Management

User_management

Role_Management

Menu management

Resource Info

add

modify

delete

Menu name	Menu description	Menu address	Sort	Menu type
1 <div>System</div>	System	/	1	1
2 <div>Information_management</div>	Basic_information_management	/	2	1
3 <div>Student_Management</div>	Student_Basic_Information_Management	studentController.do?goStudent	1	1
4 <div>Score_Management</div>	Score_information_management	scoreController.do?goScore	2	1
5 <div>Teacher_Management</div>	Teacher_Management	teacherController.do?goTeacher	3	1
6 <div>System Management</div>		/	9	1
7 <div>User_management</div>		userController.do?goUser	1	1
8 <div>Role_Management</div>		roleController.do?goRole	2	1
9 <div>Menu management</div>		resourceController.do?goResource	3	1

- Anything else that you think is relevant
- Copy CONTROLLER Source Codes and PASTE at the end of the Report package com.bjpowernode.buss.controller;

```
import java.io.InputStream;
import java.io.OutputStream;
import java.math.BigDecimal;
import java.net.URLDecoder;
import java.util.ArrayList;
import java.util.List;

import javax.servlet.http.HttpServletRequest;
```

```

import javax.servlet.http.HttpServletResponse;

import org.apache.log4j.Logger;
import org.apache.poi.hssf.usermodel.HSSFCell;
import org.apache.poi.hssf.usermodel.HSSFCellStyle;
import org.apache.poi.hssf.usermodel.HSSFRow;
import org.apache.poi.hssf.usermodel.HSSFSheet;
import org.apache.poi.hssf.usermodel.HSSFWorkbook;
import org.apache.poi.xssf.usermodel.XSSFCell;
import org.apache.poi.xssf.usermodel.XSSFRow;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
import org.hibernate.criterion.DetachedCriteria;
import org.json.JSONObject;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.multipart.MultipartFile;
import org.springframework.web.servlet.ModelAndView;

import com.bjpowernode.buss.controller.ScoreController;
import com.bjpowernode.buss.entity.base.ScoreEntity;
import com.bjpowernode.buss.entity.base.StudentEntity;
import com.bjpowernode.buss.entity.base.TeacherEntity;
import com.bjpowernode.buss.service.ScoreService;
import com.bjpowernode.common.controller.BaseController;
import com.bjpowernode.common.util.AjaxJson;
import com.bjpowernode.common.util.Pagination;

@Controller
@RequestMapping("/scoreController")
//Dependency Injection is better
//implements a common constructor and a GetContext() method, which would be
reused in all other controllers which inherit from that one
public class ScoreController extends BaseController {

    private static final Logger logger = Logger.getLogger(ScoreController.class);

    String[] excelHeader = { "StudentID", "CourseTitle", "Fraction", "Semester",
"ClassNo", "TeacherID", "TeaName", "StuName"};

```

```
@Autowired
private ScoreService scoreService;
```

```
/**
 * Initial visit
 *
 * @param
 * @param model
 * @return
 */
@RequestMapping(params = "goScore")
public ModelAndView goScore(HttpServletRequest request) {
    return new ModelAndView("buss/score");
}
```

```
@RequestMapping(params = "save")
@ResponseBody
```

```
// the ajaxJSON() function is used to perform an asynchronous request,return
JSON data
```

```
public AjaxJson save(HttpServletRequest request, HttpServletResponse
response, ScoreEntity scoreEntity,
    String studentid, String teacherid) throws Exception {
    AjaxJson j = new AjaxJson();
    j.setMsg("Saved successfully! ");
    j.setSuccess(true);
    try {
        TeacherEntity teacher =
this.scoreService.get(TeacherEntity.class, teacherid);
        StudentEntity student =
this.scoreService.get(StudentEntity.class, studentid);
        scoreEntity.setTeacherEntity(teacher);
        scoreEntity.setStudentEntity(student);
        this.scoreService.save(scoreEntity);
    } catch (Exception e) {
        j.setMsg("fail to save! ");
        j.setSuccess(false);
    }
    return j;
}
```

```
@RequestMapping(params = "update")
@ResponseBody
```

```

        public AjaxJson update(HttpServletRequest request, HttpServletResponse
response, ScoreEntity scoreEntity,
        String studentid, String teacherid) throws Exception {
            AjaxJson j = new AjaxJson();
            j.setMsg("updated sucessfully! ");
            j.setSuccess(true);
            try {
                TeacherEntity teacher =
this.scoreService.get(TeacherEntity.class, teacherid);
                StudentEntity student =
this.scoreService.get(StudentEntity.class, studentid);
                scoreEntity.setTeacherEntity(teacher);
                scoreEntity.setStudentEntity(student);
                this.scoreService.update(scoreEntity);
            } catch (Exception e) {
                j.setMsg("fail to update! ");
                j.setSuccess(false);
            }
            return j;
        }
}

```

```

@RequestMapping(params = "delete", method = RequestMethod.POST)
@ResponseBody
public AjaxJson delete(HttpServletRequest request, HttpServletResponse
response, String ids) throws Exception {
    AjaxJson j = new AjaxJson();
    j.setMsg("delete successfully! ");
    j.setSuccess(true);
    try {
        for (String id : ids.split(",")) {
            ScoreEntity scoreEntity = new ScoreEntity();
            scoreEntity.setId(id);
            this.scoreService.delete(scoreEntity);
        }
    } catch (Exception e) {
        j.setMsg("fail to delete! ");
        j.setSuccess(false);
    }
    return j;
}
}

```

//import xml file


```

@RequestMapping(params = "uploadScore", method = RequestMethod.POST)
@ResponseBody
public AjaxJson uploadScore(@RequestParam("scoreExcel") MultipartFile
scoreExcel) {
    AjaxJson j = new AjaxJson();
    j.setMsg("Imported successfully! ");
    j.setSuccess(true);
    try {
        if (!scoreExcel.isEmpty()) {
            InputStream is = scoreExcel.getInputStream();
            String[] fileName =
scoreExcel.getOriginalFilename().split("\\.");
            List<ScoreEntity> scoreList = new
ArrayList<ScoreEntity>();
            // 判断 excel 版本
            if ("xls".equals(fileName[1])) {
                // List<String> results =
                // ExcellImportUtil.getExcelStringList(is);
                HSSFWorkbook hssfWorkbook = new
HSSFWorkbook(is);
                for (int i = 0; i <
hssfWorkbook.getNumberOfSheets(); i++) {
                    HSSFSheet hssfSheet =
hssfWorkbook.getSheetAt(i);
                    if (hssfSheet == null) {
                        continue;
                    }
                    // 循环行 Row
                    for (int rowNum = 1; rowNum <=
hssfSheet.getLastRowNum(); rowNum++) {
                        HSSFRow hssfRow =
hssfSheet.getRow(rowNum);
                        if (hssfRow != null) {
                            ScoreEntity se = new
ScoreEntity();
                            TeacherEntity te = new
TeacherEntity();
                            StudentEntity stu = new
StudentEntity();
                            HSSFCell stuNum =
hssfRow.getCell(0);
                            HSSFCell courseName =

```

```

        hssfRow.getCell(2);
        hssfRow.getCell(3);
        hssfRow.getCell(4);
        hssfRow.getCell(5);
        stuNum.getNumericCellValue();
        teacherNum.getNumericCellValue();
        this.scoreService.findUniqueByProperty(TeacherEntity.class, "teachernum",
            String.valueOf(teacherInt));
        this.scoreService.findUniqueByProperty(StudentEntity.class, "studentnum",
            String.valueOf(stuInt));
        BigDecimal(score.getNumericCellValue());
        se.setTeacherEntity(te);
        se.setStudentEntity(stu);
        se.setClassname(className.getStringCellValue());
        se.setCoursename(courseName.getStringCellValue());
        se.setScore(bd);
        se.setTerm(term.getStringCellValue());
        scoreList.add(se);
    }
}
} else if ("xlsx".equals(fileName[1])) {
    XSSFWorkbook xssfWorkbook = new
XSSFWorkbook(is);
    for (int numSheet = 0; numSheet <
xssfWorkbook.getNumberOfSheets(); numSheet++) {
        XSSFSheet xssfSheet =
xssfWorkbook.getSheetAt(numSheet);
        HSSFCell score =
        HSSFCell term =
        HSSFCell className =
        HSSFCell teacherNum =
        int stuInt = (int)
        int teacherInt = (int)
        te =
        stu =
        BigDecimal bd = new
        se.setTeacherEntity(te);
        se.setScore(bd);
        scoreList.add(se);
    }
}
}
} else if ("xlsx".equals(fileName[1])) {
    XSSFWorkbook xssfWorkbook = new
XSSFWorkbook(is);
    for (int numSheet = 0; numSheet <
xssfWorkbook.getNumberOfSheets(); numSheet++) {
        XSSFSheet xssfSheet =
xssfWorkbook.getSheetAt(numSheet);

```

```

// Read the Row
for (int rowNum = 1; rowNum <=
xssfSheet.getLastRowNum(); rowNum++) {
    XSSFRow xssfRow =
    xssfSheet.getRow(rowNum);
    if (xssfRow != null) {
        ScoreEntity se = new
        TeacherEntity te = new
        StudentEntity stu = new
        XSSFCell stuNum =
        XSSFCell courseName =
        XSSFCell score =
        XSSFCell term =
        XSSFCell className =
        XSSFCell teacherNum =
        int stuInt = (int)
        int teacherInt = (int)
        te =
        this.scoreService.findUniqueByProperty(TeacherEntity.class, "teacherNum",
        String.valueOf(teacherInt));
        stu =
        this.scoreService.findUniqueByProperty(StudentEntity.class, "studentNum",
        String.valueOf(stuInt));
        BigDecimal bd = new
        BigDecimal(score.getNumericCellValue());
        se.setTeacherEntity(te);
        se.setStudentEntity(stu);
        se.setClassname(className.getStringCellValue());

```

```

        se.setCoursename(courseName.getStringCellValue());
                                                                    se.setScore(bd);

        se.setTerm(term.getStringCellValue());

                                                                    scoreList.add(se);
                                                                    }
                                                                    }
                    }
                }else{
                    j.setMsg("Please import the correct excel file!
");
                    j.setSuccess(false);
                }

                this.scoreService.saveBatch(scoreList);
            }
        } catch (Exception e) {
            e.printStackTrace();
            j.setMsg("fail to import! ");
            j.setSuccess(false);
        }
        return j;
    }

```

```

    ///export xml file
    @RequestMapping(params = "exportExcel")
    public void exportExcel(HttpServletRequest request, HttpServletResponse
response,String teachername,String coursename,
        String term,String name) throws Exception {
        DetachedCriteria condition =
DetachedCriteria.forClass(ScoreEntity.class);
        List<ScoreEntity> scoreList = this.scoreService.findData(condition,
name, teachername, coursename, term);

```

```

        HSSFWorkbook wb = new HSSFWorkbook();
        HSSFSheet sheet = wb.createSheet("sheet1");
        HSSFRow row = sheet.createRow((int) 0);
        HSSFCellStyle style = wb.createCellStyle();
        style.setAlignment(HSSFCellStyle.ALIGN_CENTER);

```

```

        for (int i = 0; i < excelHeader.length; i++) {

```

```

        HSSFCell cell = row.createCell(i);
        cell.setCellValue(excelHeader[i]);
        cell.setCellStyle(style);
        sheet.autoSizeColumn(i);
    }

    for (int i = 0; i < scoreList.size(); i++) {
        row = sheet.createRow(i + 1);
        ScoreEntity scoreEntity = scoreList.get(i);
        row.createCell(0).setCellValue(scoreEntity.getStudentEntity().getStudentnum());
        row.createCell(1).setCellValue(scoreEntity.getCoursename());
        row.createCell(2).setCellValue(scoreEntity.getScore().toString());
        row.createCell(3).setCellValue(scoreEntity.getTerm());
        row.createCell(4).setCellValue(scoreEntity.getClassname());
        row.createCell(5).setCellValue(scoreEntity.getTeacherEntity().getTeachernum());
        row.createCell(6).setCellValue(scoreEntity.getTeacherEntity().getTeachername());
        row.createCell(7).setCellValue(scoreEntity.getStudentEntity().getName());
    }

    response.setContentType("application/vnd.ms-excel");
    response.setHeader("Content-disposition",
"attachment;filename=studentScore.xls");
    OutputStream ouputStream = response.getOutputStream();
    try{
        wb.write(ouputStream);
        ouputStream.flush();
        ouputStream.close();
    }catch(Exception e){
        e.printStackTrace();
    }finally{
        ouputStream.flush();
        ouputStream.close();
    }

    }

    @RequestMapping(params = "datagrid")
    @ResponseBody
    public void datagrid(HttpServletRequest request, HttpServletResponse
response, ScoreEntity ve, String name,
        String teachername, String coursename, String term) throws
Exception {

```

号
数

```
String page = request.getParameter("page");// easyui datagrid 分页 页  
String rows = request.getParameter("rows");// easyui datagrid 分页 页  
  
if (page == null) {  
    page = "0";  
}  
if (rows == null) {  
    rows = "0";  
}  
  
DetachedCriteria condition =  
DetachedCriteria.forClass(ScoreEntity.class);  
    Pagination<?> pagination = scoreService.findPageData(condition, ve,  
Integer.parseInt(page),  
                                Integer.parseInt(rows), name, teachername,  
coursename, term);  
  
    JSONObject jobj = new JSONObject();  
    jobj.put("total", pagination.getTotalCount());  
    jobj.put("rows", pagination.getDatas());  
  
    response.setCharacterEncoding("utf-8");  
    response.getWriter().write(jobj.toString());  
  
}  
  
}  
  
package com.bjpowernode.buss.controller;  
  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
  
import org.apache.log4j.Logger;  
import org.hibernate.criterion.DetachedCriteria;  
import org.hibernate.exception.ConstraintViolationException;  
import org.json.JSONObject;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RequestMethod;  
import org.springframework.web.bind.annotation.ResponseBody;
```

```

import org.springframework.web.servlet.ModelAndView;

import com.bjpowernode.buss.controller.StudentController;
import com.bjpowernode.buss.entity.base.StudentEntity;
import com.bjpowernode.buss.service.StudentService;
import com.bjpowernode.common.controller.BaseController;
import com.bjpowernode.common.util.AjaxJson;
import com.bjpowernode.common.util.Pagination;
import com.bjpowernode.system.entity.base.UserEntity;

@Controller
@RequestMapping("/studentController")
public class StudentController extends BaseController{

    private static final Logger logger = Logger.getLogger(StudentController.class);

    @Autowired
    private StudentService studentService;

    /**
     * iniit visit
     * @param
     * @param model
     * @return
     */
    @RequestMapping(params="goStudent")
    public ModelAndView goStudent(HttpServletRequest request){
        return new ModelAndView("buss/student");
    }

    @RequestMapping(params="checkNum")
    @ResponseBody
    public void checkNum(HttpServletRequest request, HttpServletResponse
response, String studentNum) throws Exception {
        StudentEntity student =
this.studentService.findUniqueByProperty(StudentEntity.class, "studentnum",
studentNum);

        String flag = "true";
        if(student != null){
            flag = "false";
        }

        response.setCharacterEncoding("utf-8");
        response.getWriter().write(flag);
    }
}

```

```

    }

    @RequestMapping(params="save")
    @ResponseBody
    public AjaxJson save(HttpServletRequest request, HttpServletResponse
response, StudentEntity student) throws Exception {
        AjaxJson j = new AjaxJson();
        j.setMsg("saved sucessfully! ");
        j.setSuccess(true);
        try{
            this.studentService.save(student);
        }catch(Exception e){
            j.setMsg("fail to save! ");
            j.setSuccess(false);
        }
        return j;
    }

    @RequestMapping(params="update")
    @ResponseBody
    public AjaxJson update(HttpServletRequest request, HttpServletResponse
response, StudentEntity student) throws Exception {
        AjaxJson j = new AjaxJson();
        j.setMsg("updated sucessfully! ");
        j.setSuccess(true);
        try{
//            StudentEntity se = this.studentService.get(StudentEntity.class,
student.getId());

            this.studentService.update(student);
        }catch(Exception e){
            j.setMsg("fail to update! ");
            j.setSuccess(false);
        }
        return j;
    }

    @RequestMapping(params="delete",method=RequestMethod.POST)
    @ResponseBody
    public AjaxJson delete(HttpServletRequest request, HttpServletResponse
response, String ids) throws Exception {

```



```

        AjaxJson j = new AjaxJson();
        j.setMsg("delete successfully! ");
        j.setSuccess(true);
        try{
            for(String id:ids.split(", ")){
                StudentEntity student =
this.studentService.get(StudentEntity.class, id);
                this.studentService.delete(student);
            }
        }catch(ConstraintViolationException ce){
            ce.printStackTrace();
            j.setMsg("fail to delete,there exist foreign key references,
Please check if there is any information related to the current data in other data
items!");
            j.setSuccess(false);
        }catch(Exception e){
            j.setMsg("fail to delete! ");
            j.setSuccess(false);
        }
        return j;
    }
}

```

```

    @RequestMapping(params="datagrid")
    @ResponseBody
    public void datagrid(HttpServletRequest request, HttpServletResponse
response,StudentEntity student) throws Exception {
        String page = request.getParameter("page");
        String rows = request.getParameter("rows");
        if(page == null){
            page = "0";
        }
        if(rows == null){
            rows = "0";
        }
        DetachedCriteria condition =
DetachedCriteria.forClass(StudentEntity.class);
        Pagination<?> pagination =
studentService.findPageData(condition,student,Integer.parseInt(page),
Integer.parseInt(rows));
        JSONObject jobj = new JSONObject();
        jobj.put("total", pagination.getTotalCount());
        jobj.put("rows", pagination.getDatas());
    }
}

```

```
        response.setCharacterEncoding("utf-8");
        response.getWriter().write(jobj.toString());

    }

}
```

```
package com.bjpowernode.buss.controller;
```

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

```
import org.apache.log4j.Logger;
import org.hibernate.criterion.DetachedCriteria;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.servlet.ModelAndView;
```

```
import com.bjpowernode.buss.controller.TeacherController;
import com.bjpowernode.buss.entity.base.StudentEntity;
import com.bjpowernode.buss.entity.base.TeacherEntity;
import com.bjpowernode.buss.service.TeacherService;
import com.bjpowernode.common.controller.BaseController;
import com.bjpowernode.common.util.AjaxJson;
import com.bjpowernode.common.util.Pagination;
```

```
import org.json.JSONObject;
```

```
@Controller
@RequestMapping("/teacherController")
public class TeacherController extends BaseController{
```

```
    private static final Logger logger = Logger.getLogger(TeacherController.class);
```

```
    @Autowired
    private TeacherService teacherService;
```

```
    /**
     * init visit
     * @param
```

```

        * @param model
        * @return
        */
        @RequestMapping(params="goTeacher")
        public ModelAndView goTeacher(HttpServletRequest request){
            return new ModelAndView("buss/teacher");
        }

        @RequestMapping(params="checkTeachernum")
        @ResponseBody
        public void checkTeachernum(HttpServletRequest request, HttpServletResponse
response, String teacherNum) throws Exception {
            TeacherEntity teacher =
this.teacherService.findUniqueByProperty(TeacherEntity.class, "teachernum",
teacherNum);

            String flag = "true";
            if(teacher != null){
                flag = "false";
            }

            response.setCharacterEncoding("utf-8");
            response.getWriter().write(flag);

        }

        @RequestMapping(params="save")
        @ResponseBody
        public AjaxJson save(HttpServletRequest request, HttpServletResponse
response, TeacherEntity teacherEntity) throws Exception {
            AjaxJson j = new AjaxJson();
            j.setMsg("saved successfully! ");
            j.setSuccess(true);
            try{
                this.teacherService.save(teacherEntity);
            }catch(Exception e){
                j.setMsg("fail to save! ");
                j.setSuccess(false);
            }
            return j;
        }

        @RequestMapping(params="update")
        @ResponseBody

```

```

        public AjaxJson update(HttpServletRequest request, HttpServletResponse
response, TeacherEntity teacherEntity) throws Exception {
            AjaxJson j = new AjaxJson();
            j.setMsg("updated successfully! ");
            j.setSuccess(true);
            try{
                this.teacherService.update(teacherEntity);
            }catch(Exception e){
                j.setMsg("fail to update! ");
                j.setSuccess(false);
            }
            return j;
        }
    }

```

```

        @RequestMapping(params="delete",method=RequestMethod.POST)
        @ResponseBody
        public AjaxJson delete(HttpServletRequest request, HttpServletResponse
response, String ids) throws Exception {
            AjaxJson j = new AjaxJson();
            j.setMsg("delete successfully! ");
            j.setSuccess(true);
            try{
                for(String id:ids.split(",")){
                    TeacherEntity teacherEntity = new TeacherEntity();
                    teacherEntity = teacherService.get(TeacherEntity.class,
id);

                    this.teacherService.delete(teacherEntity);
                }
            }catch(Exception e){
                j.setMsg("fail to dalete! ");
                j.setSuccess(false);
            }
            return j;
        }
    }

```

```

        @RequestMapping(params="datagrid")
        @ResponseBody
        public void datagrid(HttpServletRequest request, HttpServletResponse
response, TeacherEntity ve) throws Exception {
            String page = request.getParameter("page");//easyui datagrid 分页 页
号

```

```

        String rows = request.getParameter("rows");//easyui datagrid 分页 页
数
        if(page == null){
            page = "0";
        }
        if(rows == null){
            rows = "0";
        }
        DetachedCriteria condition =
DetachedCriteria.forClass(TeacherEntity.class);
        Pagination<?> pagination =
teacherService.findPageData(condition,ve,Integer.parseInt(page),
Integer.parseInt(rows));

        JSONObject jobj = new JSONObject();
        jobj.put("total", pagination.getTotalCount());
        jobj.put("rows", pagination.getDatas());

        response.setCharacterEncoding("utf-8");
        response.getWriter().write(jobj.toString());

    }

}

```

```

package com.bjpowernode.common.controller;

import java.util.Date;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.ServletRequestDataBinder;
import org.springframework.web.bind.annotation.InitBinder;

import com.alibaba.fastjson.JSON;
import com.alibaba.fastjson.serializer.SerializerFeature;
import com.bjpowernode.common.util.DateConvertEditor;

/**
 *
 * jiaqi Wang
 * Package basic content(controller)
 */
@Controller

```

```

public class BaseController {

    /**
     * The date format string passed from the front desk is automatically converted
to Date type
     *
     * @param binder
     */
    @InitBinder
    public void initBinder(ServletRequestDataBinder binder) {
        binder.registerCustomEditor(Date.class, new DateConvertEditor());
    }

    // public BaseController(){
    //     JSON.DEFAULT_GENERATE_FEATURE |=
    //     SerializerFeature.DisableCircularReferenceDetect.getMask();
    // }
}

```

```

package com.bjpowernode.system.controller;

```

```

import java.util.ArrayList;
import java.util.Collections;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

```

```

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

```

```

import org.apache.log4j.Logger;
import org.json.JSONObject;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.view.RedirectView;

```

```

import com.bjpowernode.common.util.AjaxJson;
import com.bjpowernode.common.util.ContextHolderUtils;

```

```

import com.bjpowernode.common.util.ResourceUtil;
import com.bjpowernode.common.util.SystemConstant;
import com.bjpowernode.system.controller.LoginController;
import com.bjpowernode.system.entity.base.ResourceEntity;
import com.bjpowernode.system.entity.base.RoleEntity;
import com.bjpowernode.system.entity.base.UserEntity;
import com.bjpowernode.system.manager.ClientManager;
import com.bjpowernode.system.service.SystemService;
import com.bjpowernode.system.vo.Client;
import com.bjpowernode.system.vo.TreeNode;

@Controller
@RequestMapping("/loginController")
public class LoginController {

    private static final Logger logger = Logger.getLogger(LoginController.class);

    @Autowired
    private SystemService systemService;

    /**
     * Login page
     * @param error
     * @param model
     * @return
     */
    @RequestMapping(params="login")
    public ModelAndView login(HttpServletRequest request){
        return new ModelAndView("system/login");
    }

    @RequestMapping(params="home")
    public ModelAndView home(HttpServletRequest request){
        return new ModelAndView("system/home");
    }

    /**
     * Exit system
     *
     * @param user
     * @param req
     * @return
     */
    @RequestMapping(params = "logout")

```

```

        public ModelAndView logout(HttpServletRequest request, HttpServletResponse
response) {
            HttpSession session = ContextHolderUtils.getSession();
            ClientManager.getInstance().removeClient(session.getId());
            session.invalidate();
            ModelAndView modelAndView = new ModelAndView(new
RedirectView(
                "loginController.do?login"));

            return modelAndView;
        }

```

```

        @RequestMapping(params="doLogin")
        public ModelAndView doLogin(HttpServletRequest req){
            ModelAndView mav = new ModelAndView("system/main");
            HttpSession session = ContextHolderUtils.getSession();
            Client client = ClientManager.getInstance().getClient(session.getId());
            req.setAttribute("username", client.getUser().getUsername());
            return mav;
        }

```

```

        @RequestMapping(params="doCheck")
        @ResponseBody
        public AjaxJson doCheck(HttpServletRequest req,String username, String password,
String captcha){
            HttpSession session = ContextHolderUtils.getSession();
            AjaxJson j = new AjaxJson();

```

```

            if(!captcha.equalsIgnoreCase(String.valueOf(session.getAttribute(SystemConsta
nt.KEY_CAPTCHA)))){
                j.setSuccess(false);
                j.setMsg("Verification code error!");
            }else{
                UserEntity user = new UserEntity();
                user.setUsername(username);
                user.setPassword(password);
                user = this.systemService.getUserByNameAndPassword(user);
                if(user == null){
                    j.setSuccess(false);
                    j.setMsg("wrong user name or password! ");
                    return j;
                }
            }
        }

```



```

        List<ResourceEntity> resourceList = new
ArrayList<ResourceEntity>();
        List<RoleEntity> roleList = user.getRoles();
        for(RoleEntity re : roleList){
            List<ResourceEntity> tempRes = re.getResource();
            for(ResourceEntity res : tempRes){
                if(!resourceList.contains(res)){
                    resourceList.add(res);
                }
            }
        }
        Client client = new Client();
        client.setIp(ResourceUtil.getIpAddr(req));
        client.setLogindatetime(new Date());
        client.setUser(user);
        client.setMenuList(resourceList);
        ClientManager.getInstance().addClient(session.getId(),
            client);
        if(user != null && user.getId() != null){
            if(user.getStatus() == 2){
                j.setSuccess(false);
                j.setMsg("This user is disabled, please contact
the administrator! ");
            }else{
                j.setSuccess(true);
                j.setMsg("Sign in successfully! ");
            }
        }else{
            j.setSuccess(false);
            j.setMsg("wrong user name or password!");
        }
    }
    return j;
}

```

```

@RequestMapping(params="getTreeMenu")
@ResponseBody
public String getTreeMenu(HttpServletRequest request){
    Client client = ResourceUtil.getClient();
    List<ResourceEntity> resourceList = new ArrayList<ResourceEntity>();
    if(client == null || client.getUser() == null){

```

```

        return "system/login";
    }else{
        resourceList = client.getMenuList();
    }

    List<ResourceEntity> resource = new ArrayList<ResourceEntity>();
    for(ResourceEntity re:resourceList){
        if(resourceList.size()<=0){
            break;
        }
        if("1".equals(re.getId())){
            resource.add(re);
            break;
        }
    }
    return
JSONObject.valueToString(resourceToTreeNode(resource,resourceList));
}

/**
 * 将 sysResource 类型的数据集合转化为前端较好识别的 TreeNode
 * @param resource
 * @return
 */
private List<TreeNode> resourceToTreeNode(List<ResourceEntity> resource,
List<ResourceEntity> userResource) {
    if (resource != null && !resource.isEmpty() &&
resource.get(0).getResourceType() == ResourceEntity.TYPE_MENU) {
        List<TreeNode> ch = new ArrayList<TreeNode>();
        for (ResourceEntity rr : resource) {
            TreeNode node = new TreeNode();
            if(userResource.contains(rr)){
                if(rr.getHref()==null){
                    node.setId(rr.getId());
                }else{
                    node.setId(rr.getId());
                }
            }
            node.setId(rr.getId());
            node.setState("open");
            node.setText(rr.getName());

            Map<String, Object> _temp = new
HashMap<String, Object>();

            _temp.put("href", rr.getHref());

```

```

        node.setAttributes(_temp);

        ch.add(node);

    }

    node.setChildren(resourceToTreeNode(rr.getResources(),userResource));
    }

    return ch;
}
return Collections.emptyList();
}
}

```

```
package com.bjpowernode.system.controller;
```

```
import java.util.List;
```

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

```
import org.apache.log4j.Logger;
import org.hibernate.criterion.DetachedCriteria;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.servlet.ModelAndView;
```

```
import com.bjpowernode.common.controller.BaseController;
import com.bjpowernode.common.util.AjaxJson;
import com.bjpowernode.common.util.Pagination;
import com.bjpowernode.system.controller.ResourceController;
import com.bjpowernode.system.entity.base.ResourceEntity;
import com.bjpowernode.system.service.SystemService;
```

```
@Controller
@RequestMapping("/resourceController")
public class ResourceController extends BaseController{
```

```
    private static final Logger logger = Logger.getLogger(ResourceController.class);
```

```

    @Autowired
    private SystemService systemService;

    /**
     * Landing page
     * @param error
     * @param model
     * @return
     */
    @RequestMapping(params="goResource")
    public ModelAndView goResource(HttpServletRequest request){
        return new ModelAndView("system/resource");
    }

    @RequestMapping(params="save")
    @ResponseBody
    public AjaxJson save(HttpServletRequest request, HttpServletResponse
response, ResourceEntity resource,String parentid) throws Exception {
        AjaxJson j = new AjaxJson();
        j.setMsg("saved successfully! ");
        j.setSuccess(true);
        try{
            ResourceEntity patentRes = new ResourceEntity();
            patentRes.setld(parentid);
            resource.setParentResource(patentRes);
            this.systemService.save(resource);
        }catch(Exception e){
            j.setMsg("fail to save! ");
            j.setSuccess(false);
        }
        return j;
    }

    @RequestMapping(params="update")
    @ResponseBody
    public AjaxJson update(HttpServletRequest request, HttpServletResponse
response, ResourceEntity resource, String parentid) throws Exception {
        AjaxJson j = new AjaxJson();
        j.setMsg("update successfully! ");
        j.setSuccess(true);
        try{

```

```

        ResourceEntity patentRes = new ResourceEntity();
        patentRes.setId(parentid);
        resource.setParentResource(patentRes);
        this.systemService.update(resource);
    }catch(Exception e){
        j.setMsg("fail to uodate! ");
        j.setSuccess(false);
    }
    return j;
}

```

`@RequestMapping(params="delete",method=RequestMethod.POST)`
 //tells a controller that the object returned is automatically serialized into JSON
 and passed back into the `HttpResponse` object.

```

    @ResponseBody
    public AjaxJson delete(HttpServletRequest request, HttpServletResponse
response, String ids) throws Exception {
        AjaxJson j = new AjaxJson();
        j.setMsg("delete successfully! ");
        j.setSuccess(true);
        try{
            for(String id:ids.split(", ")){
                ResourceEntity resource = new ResourceEntity();
                resource.setId(id);
                this.systemService.delete(resource);
            }
        }catch(Exception e){
            j.setMsg("fail to delete! ");
            j.setSuccess(false);
        }
        return j;
    }
}

```

```

    @RequestMapping(params="datagrid")
    @ResponseBody
    public void datagrid(HttpServletRequest request, HttpServletResponse
response) throws Exception {
        String page = request.getParameter("page");//easyui datagrid 分页 页
号
        String rows = request.getParameter("rows");//easyui datagrid 分页 页
数
        if(page == null){

```

```

        page = "0";
    }
    if(rows == null){
        rows = "0";
    }
    DetachedCriteria condition =
DetachedCriteria.forClass(ResourceEntity.class);
    Pagination<?> pagination =
systemService.getPageData(condition,Integer.parseInt(page), Integer.parseInt(rows));
    List<ResourceEntity> list= (List<ResourceEntity>)pagination.getDatas();
    String resourceJson = systemService.getTreeJson(list);

    response.setCharacterEncoding("utf-8");//指定为 utf-8
    response.getWriter().write(resourceJson);//转化为 JSON 格式

}

@RequestMapping(params="treeDropdown")
@ResponseBody
public void treeDropdown(HttpServletRequest request, HttpServletResponse
response) throws Exception {
    String page = request.getParameter("page");
    String rows = request.getParameter("rows");
    if(page == null){
        page = "0";
    }
    if(rows == null){
        rows = "0";
    }
    DetachedCriteria condition =
DetachedCriteria.forClass(ResourceEntity.class);
    Pagination<?> pagination =
systemService.getPageData(condition,Integer.parseInt(page), Integer.parseInt(rows));
    List<ResourceEntity> list= (List<ResourceEntity>)pagination.getDatas();
    String retJson = systemService.getTreeJson(list);
    String resourceJson = retJson.replaceAll("\"name\\\"", "\"text\\\"");
    response.setCharacterEncoding("utf-8");
    response.getWriter().write(resourceJson);

}
}

```

```

package com.bjpowernode.system.controller;

```

```

import java.util.ArrayList;
import java.util.List;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.log4j.Logger;
import org.hibernate.criterion.DetachedCriteria;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.servlet.ModelAndView;

import com.alibaba.fastjson.JSONObject;
import com.bjpowernode.common.controller.BaseController;
import com.bjpowernode.common.util.AjaxJson;
import com.bjpowernode.common.util.Pagination;
import com.bjpowernode.system.controller.RoleController;
import com.bjpowernode.system.entity.base.ResourceEntity;
import com.bjpowernode.system.entity.base.RoleEntity;
import com.bjpowernode.system.service.SystemService;

@Controller
@RequestMapping("/roleController")
public class RoleController extends BaseController{

    private static final Logger logger = Logger.getLogger(RoleController.class);

    @Autowired
    private SystemService systemService;

    /**
     * login page
     * @param error
     * @param model
     * @return
     */
    @RequestMapping(params="goRole")
    public ModelAndView goRole(HttpServletRequest request){
        return new ModelAndView("system/role");
    }

```

```
}
```

```
private List<ResourceEntity> getResources(String resourceids){  
    if(resourceids == null){  
        return null;  
    }  
    String[] resources = resourceids.split(",");  
    List<ResourceEntity> resList = new ArrayList<ResourceEntity>();  
    for(int i=0; i<resources.length; i++){  
        ResourceEntity res = new ResourceEntity();  
        res = this.systemService.get(ResourceEntity.class, resources[i]);  
        resList.add(res);  
    }  
    return resList;  
}
```

```
    @RequestMapping(params="save")  
    @ResponseBody  
    public AjaxJson save(HttpServletRequest request, HttpServletResponse  
response, RoleEntity role,String resourceids) throws Exception {  
        AjaxJson j = new AjaxJson();  
  
        role.setResource(getResources(resourceids));  
        j.setMsg("saved successfully! ");  
        j.setSuccess(true);  
        try{  
            this.systemService.save(role);  
        }catch(Exception e){  
            j.setMsg("fail to save! ");  
            j.setSuccess(false);  
        }  
        return j;  
    }  
}
```

```
    @RequestMapping(params="update")  
    @ResponseBody  
    public AjaxJson update(HttpServletRequest request, HttpServletResponse  
response, RoleEntity role, String resourceids) throws Exception {  
        AjaxJson j = new AjaxJson();  
        j.setMsg("updated successfully! ");  
        j.setSuccess(true);  
        try{  
            role.setResource(getResources(resourceids));  
        }  
    }  
}
```



```

        this.systemService.update(role);
    }catch(Exception e){
        j.setMsg("fail to update! ");
        j.setSuccess(false);
    }
    return j;
}

@RequestMapping(params="delete",method=RequestMethod.POST)
@ResponseBody
public AjaxJson delete(HttpServletRequest request, HttpServletResponse
response, String ids) throws Exception {
    AjaxJson j = new AjaxJson();
    j.setMsg("delete successfully! ");
    j.setSuccess(true);
    try{
        for(String id:ids.split(",")){
            RoleEntity role = new RoleEntity();
            role.setId(id);
            this.systemService.delete(role);
        }
    }catch(Exception e){
        j.setMsg("fail to delete! ");
        j.setSuccess(false);
    }
    return j;
}

@RequestMapping(params="queryResource",method=RequestMethod.POST)
@ResponseBody
public AjaxJson queryResource(HttpServletRequest request,
HttpServletResponse response, String id) throws Exception {
    AjaxJson j = new AjaxJson();
    j.setMsg("success! ");
    j.setSuccess(true);
    try{
        RoleEntity re = this.systemService.get(RoleEntity.class, id);
        String resourceId = "";
        for(ResourceEntity res:re.getResource()){
            if(res.getResources() == null || res.getResources().size()
== 0){
                resourceId += (res.getId() + ",");
            }
        }
    }
}

```

```

        }

        }
        if(resourceId.length()>0){
            resourceId = resourceId.substring(0,
resourceId.length()-1);
        }
        j.setObj(resourceId);
    }catch(Exception e){
        j.setMsg("failure! ");
        j.setSuccess(false);
    }
    return j;
}

    }

    @RequestMapping(params="dropdown")
    @ResponseBody
    public void dropdown(HttpServletRequest request, HttpServletResponse
response) throws Exception {
        DetachedCriteria condition =
DetachedCriteria.forClass(RoleEntity.class);
        Pagination<?> pagination = systemService.getPageData(condition,0, 0);
        List<RoleEntity> list= (List<RoleEntity>)pagination.getDatas();
        StringBuffer sb = new StringBuffer();
        sb.append("[");
        for(RoleEntity re : list){
            sb.append("{");
            sb.append("\"id\":");
            sb.append("\"");
            sb.append(re.getId());
            sb.append("\"");
            sb.append(",");
            sb.append("\"text\":");
            sb.append("\"");
            sb.append(re.getName());
            sb.append("\"");
            sb.append("},");
        }
        String dropdown = sb.substring(0, sb.length()-1);
        dropdown += "]";
        response.setCharacterEncoding("utf-8");//指定为 utf-8
        response.getWriter().write(dropdown);//转化为 JSON 格式
    }
}

```

```

    }

    @RequestMapping(params="datagrid")
    @ResponseBody
    public void datagrid(HttpServletRequest request, HttpServletResponse
response) throws Exception {
        String page = request.getParameter("page");//easyui datagrid 分页 页
号
        String rows = request.getParameter("rows");//easyui datagrid 分页 页
数

        if(page == null){
            page = "0";
        }
        if(rows == null){
            rows = "0";
        }
        DetachedCriteria condition =
DetachedCriteria.forClass(RoleEntity.class);
        Pagination<?> pagination =
systemService.getPageData(condition,Integer.parseInt(page), Integer.parseInt(rows));
        JSONObject jobj = new JSONObject();
        jobj.put("total", pagination.getTotalCount());
        jobj.put("rows", pagination.getDatas());

        response.setCharacterEncoding("utf-8");
        response.getWriter().write(jobj.toString());

    }
}

```

```

package com.bjpowernode.system.controller;

import java.util.ArrayList;
import java.util.List;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.commons.beanutils.BeanUtils;
import org.apache.log4j.Logger;
import org.apache.shiro.crypto.hash.Md5Hash;
import org.hibernate.criterion.DetachedCriteria;

```

```

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.servlet.ModelAndView;

import com.alibaba.fastjson.JSONObject;
import com.bjpowernode.common.controller.BaseController;
import com.bjpowernode.common.util.AjaxJson;
import com.bjpowernode.common.util.Pagination;
import com.bjpowernode.system.controller.UserController;
import com.bjpowernode.system.entity.base.RoleEntity;
import com.bjpowernode.system.entity.base.UserEntity;
import com.bjpowernode.system.service.SystemService;

@Controller
@RequestMapping("/userController")
public class UserController extends BaseController{

    private static final Logger logger = Logger.getLogger(UserController.class);

    @Autowired
    private SystemService systemService;

    /**
     * login page
     * @param
     * @param model
     * @return
     */
    @RequestMapping(params="goUser")
    public ModelAndView goUser(HttpServletRequest request){
        return new ModelAndView("system/user");
    }

    @RequestMapping(params="checkRemote")
    @ResponseBody
    public void checkRemote(HttpServletRequest request, HttpServletResponse
response, String signcode) throws Exception {
        UserEntity user =
this.systemService.findUniqueByProperty(UserEntity.class, "username", signcode);
        String flag = "true";

```

```

        if(user != null){
            flag = "false";
        }
        response.setCharacterEncoding("utf-8");
        response.getWriter().write(flag);
    }

    @RequestMapping(params="save")
    @ResponseBody
    public AjaxJson save(HttpServletRequest request, HttpServletResponse
response, UserEntity user, String roleid) throws Exception {
        AjaxJson j = new AjaxJson();
        j.setMsg("saved successfully! ");
        j.setSuccess(true);
        try{
            Md5Hash md5Hash = new Md5Hash(user.getPassword());
            user.setPassword(md5Hash.toHex());
            user.setRoles(getRoleList(roleid));
            this.systemService.save(user);
        }catch(Exception e){
            j.setMsg("fail to save! ");
            j.setSuccess(false);
        }
        return j;
    }

    @RequestMapping(params="update")
    @ResponseBody
    public AjaxJson update(HttpServletRequest request, HttpServletResponse
response, UserEntity user, String roleid) throws Exception {
        AjaxJson j = new AjaxJson();
        j.setMsg("updated successfully! ");
        j.setSuccess(true);
        try{
            UserEntity t = this.systemService.get(UserEntity.class,
user.getId());
            user.setRoles(getRoleList(roleid));
            user.setPassword(t.getPassword());
            user.setUsername(t.getUsername());
            BeanUtils.copyProperties(t, user);
            this.systemService.update(t);
        }catch(Exception e){

```

```

        j.setMsg("fail to update! ");
        j.setSuccess(false);
    }
    return j;
}

private List<RoleEntity> getRoleList(String roleid){
    if(roleid == null){
        return null;
    }
    String[] ids = roleid.split(",");
    List<RoleEntity> roleList = new ArrayList<RoleEntity>();
    for(String id:ids){
        RoleEntity re = this.systemService.get(RoleEntity.class, id);
        roleList.add(re);
    }
    return roleList;
}

@RequestMapping(params="delete",method=RequestMethod.POST)
@ResponseBody
public AjaxJson delete(HttpServletRequest request, HttpServletResponse
response, String ids) throws Exception {
    AjaxJson j = new AjaxJson();
    j.setMsg("delete successfully! ");
    j.setSuccess(true);
    try{
        for(String id:ids.split(",")){
            UserEntity user = new UserEntity();
            user.setId(id);
            this.systemService.delete(user);
        }
    }catch(Exception e){
        j.setMsg("fail to delete! ");
        j.setSuccess(false);
    }
    return j;
}

@RequestMapping(params="queryRole",method=RequestMethod.POST)
@ResponseBody

```

```

        public AjaxJson queryRole(HttpServletRequest request, HttpServletResponse
response, String id) throws Exception {
            AjaxJson j = new AjaxJson();
            j.setMsg("success! ");
            j.setSuccess(true);
            try{
                UserEntity user = this.systemService.get(UserEntity.class, id);
                String roleId = "";
                for(RoleEntity re:user.getRoles()){
                    roleId += (re.getId() + ",");
                }
                if(roleId.length()>0){
                    roleId = roleId.substring(0, roleId.length()-1);
                }
                j.setObj(roleId);
            }catch(Exception e){
                j.setMsg("failure! ");
                j.setSuccess(false);
            }
            return j;
        }
    }

```

```

        @RequestMapping(params="datagrid")
        @ResponseBody
        public void datagrid(HttpServletRequest request, HttpServletResponse
response) throws Exception {
            String page = request.getParameter("page");//easyui datagrid 分页 页
            号

            String rows = request.getParameter("rows");//easyui datagrid 分页 页
            数

            if(page == null){
                page = "0";
            }
            if(rows == null){
                rows = "0";
            }
            DetachedCriteria condition =
DetachedCriteria.forClass(UserEntity.class);
            Pagination<?> pagination =
systemService.getPageData(condition,Integer.parseInt(page), Integer.parseInt(rows));

            JSONObject jobj = new JSONObject();
            jobj.put("total", pagination.getTotalCount());

```

```

        jobj.put("rows", pagination.getDatas());

        response.setCharacterEncoding("utf-8");
        response.getWriter().write(jobj.toString());

    }
}

```

- Copy DAO Source Codes and PASTE at the end of the Report package com.bjpowernode.system.dao;

```

import java.util.List;

import com.bjpowernode.common.dao.BaseDao;
import com.bjpowernode.system.entity.base.ResourceEntity;
import com.bjpowernode.system.entity.base.UserEntity;

public interface SystemDao extends BaseDao {

    public UserEntity getUserByNameAndPassword(UserEntity user);

    public List<ResourceEntity> getTreeMenuResource(UserEntity user);

    /**
     * Get unique records based on entity name
     *
     * @param propertyName
     * @param value
     * @return
     */
    public <T> T findUniqueByProperty(Class<T> entityClass,
                                     String propertyName, Object value);
}

```

```

package com.bjpowernode.system.dao.impl;

```

```

import java.util.List;

```

```

import org.apache.shiro.crypto.hash.Md5Hash;
import org.hibernate.Criteria;
import org.hibernate.Query;
import org.hibernate.criterion.Criterion;
import org.hibernate.criterion.Restrictions;

```



```

import org.springframework.stereotype.Repository;

import com.bjpowernode.common.dao.impl.BaseDaoImpl;
import com.bjpowernode.system.dao.SystemDao;
import com.bjpowernode.system.entity.base.ResourceEntity;
import com.bjpowernode.system.entity.base.UserEntity;

@Repository
public class SystemDaoImpl extends BaseDaoImpl implements
SystemDao {

    @Override
    public UserEntity getUserByNameAndPassword(UserEntity user)
    {
        Md5Hash md5Hash = new Md5Hash(user.getPassword());
        String password = md5Hash.toHex();
        String query = "from UserEntity u where u.username
= :username and u.password=:passowrd";
        Query queryObject = getSession().createQuery(query);
        queryObject.setParameter("username",
user.getUsername());
        queryObject.setParameter("passowrd", password);
        List<UserEntity> users = queryObject.list();
        if (users != null && users.size() > 0) {
            return users.get(0);
        }

        return null;
    }

    /**
     * 获取用户左侧权限菜单
     */
    @Override
    public List<ResourceEntity> getTreeMenuResource(UserEntity
user) {
        String hql = "select r.resource from UserEntity u
inner join fecth u.roles r where u.id = :id";
        Query queryObject = getSession().createQuery(hql);
        queryObject.setParameter("id", user.getId());
        List<ResourceEntity> resourceList =
queryObject.list();

        return resourceList;
    }
}

```

```

        public <T> T findUniqueByProperty(Class<T> entityClass,
            String propertyName, Object value) {
            return (T) createCriteria(entityClass,
                Restrictions.eq(propertyName,
value)).uniqueResult();
        }

        private <T> Criteria createCriteria(Class<T> entityClass,
            Criterion... criterions) {
            Criteria criteria =
getSession().createCriteria(entityClass);
            for (Criterion c : criterions) {
                criteria.add(c);
            }
            return criteria;
        }
    }

package com.bjpowernode.system.dao;

import java.util.List;

import com.bjpowernode.common.dao.BaseDao;
import com.bjpowernode.system.entity.base.ResourceEntity;
import com.bjpowernode.system.entity.base.UserEntity;

public interface SystemDao extends BaseDao {

    public UserEntity getUserByNameAndPassword(UserEntity user);

    public List<ResourceEntity> getTreeMenuResource(UserEntity
user);

    /**
     * Get unique records based on entity name
     *
     * @param propertyName
     * @param value
     * @return
     */
    public <T> T findUniqueByProperty(Class<T> entityClass,
        String propertyName, Object value);
}

```

```

package com.bjpowernode.system.dao.impl;

import java.util.List;

import org.apache.shiro.crypto.hash.Md5Hash;
import org.hibernate.Criteria;
import org.hibernate.Query;
import org.hibernate.criterion.Criterion;
import org.hibernate.criterion.Restrictions;
import org.springframework.stereotype.Repository;

import com.bjpowernode.common.dao.impl.BaseDaoImpl;
import com.bjpowernode.system.dao.SystemDao;
import com.bjpowernode.system.entity.base.ResourceEntity;
import com.bjpowernode.system.entity.base.UserEntity;

@Repository
public class SystemDaoImpl extends BaseDaoImpl implements
SystemDao {

    @Override
    public UserEntity getUserByNameAndPassword(UserEntity user)
    {
        Md5Hash md5Hash = new Md5Hash(user.getPassword());
        String password = md5Hash.toHex();
        String query = "from UserEntity u where u.username
= :username and u.password=:passowrd";
        Query queryObject = getSession().createQuery(query);
        queryObject.setParameter("username",
user.getUsername());
        queryObject.setParameter("passowrd", password);
        List<UserEntity> users = queryObject.list();
        if (users != null && users.size() > 0) {
            return users.get(0);
        }

        return null;
    }

    /**
     * 获取用户左侧权限菜单
     */
    @Override

```

```

    public List<ResourceEntity> getTreeMenuResource(UserEntity
user) {
        String hql = "select r.resource from UserEntity u
inner join fetch u.roles r where u.id = :id";
        Query queryObject = getSession().createQuery(hql);
        queryObject.setParameter("id", user.getId());
        List<ResourceEntity> resourceList =
queryObject.list();

        return resourceList;
    }

    public <T> T findUniqueByProperty(Class<T> entityClass,
String propertyName, Object value) {
        return (T) createCriteria(entityClass,
Restrictions.eq(propertyName,
value)).uniqueResult();
    }

    private <T> Criteria createCriteria(Class<T> entityClass,
Criterion... criterions) {
        Criteria criteria =
getSession().createCriteria(entityClass);
        for (Criterion c : criterions) {
            criteria.add(c);
        }
        return criteria;
    }
}

```