医院挂号系统实验报告

一、实验目的与要求

1. 掌握 java web 开发环境的搭建

2. 掌握 java web 项目的创建、发布和运行

3. 理解 MVC 的开发模式

4. 掌握基础的多角色系统设计与实现

二、实验内容

病人挂号系统

要求：记录实验步骤及实验结果截图，填写实验报告，提交程序文件夹

请完成以下内容：

1）实现不同类型用户的注册与登陆（在之前实验的基础上将医生与病人作为系统用户），使用 Servlet+JSP+JDBC 或者其他框架技术，并设计相应的 mysql 数据库。

2）用户登录成功之后进入主页，把用户的用户名存入 session 并显示在页面上。

3）提前设置好一些科室，医生可以维护自己所属科室信息。

4）病人登陆跳转至挂号页显示自己的挂号记录，并可以选择科室进行挂号。

5）当病人提交挂号信息，**按不同时间段计算并记录挂号费**。

6）医生登陆跳转至记录页显示自己所在科室的挂号记录，并可以根据不同维度对挂号记录进行筛选。

附加实验 挂号系统扩展

要求：在实验六的基础上进行扩展

【实验目的】

1.掌握 JavaMail 发送邮件

2.掌握 poi 生成报表

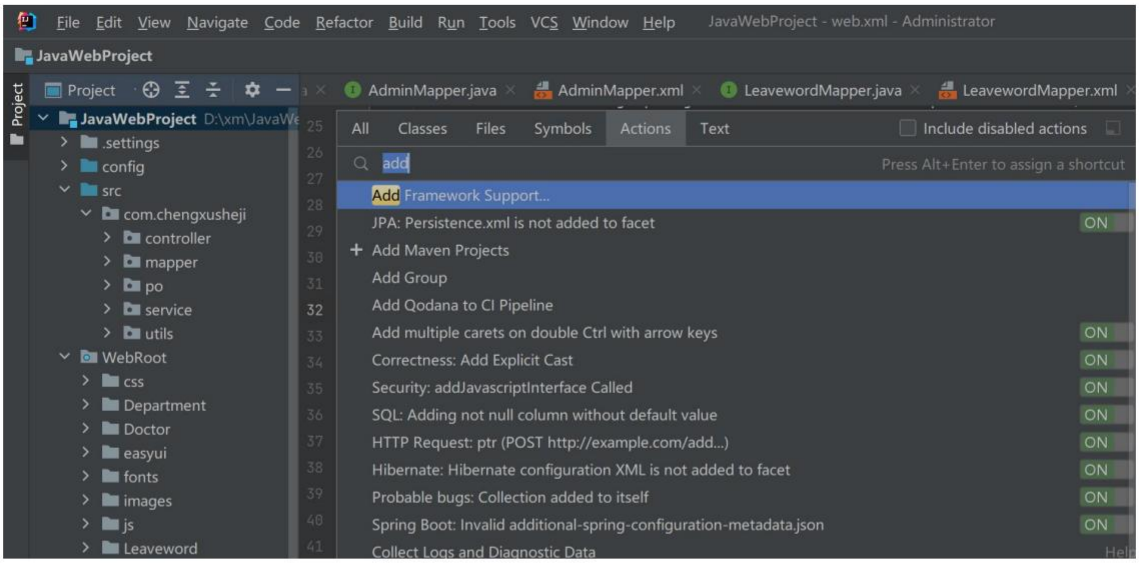
三、实验步骤及其结果截图展示

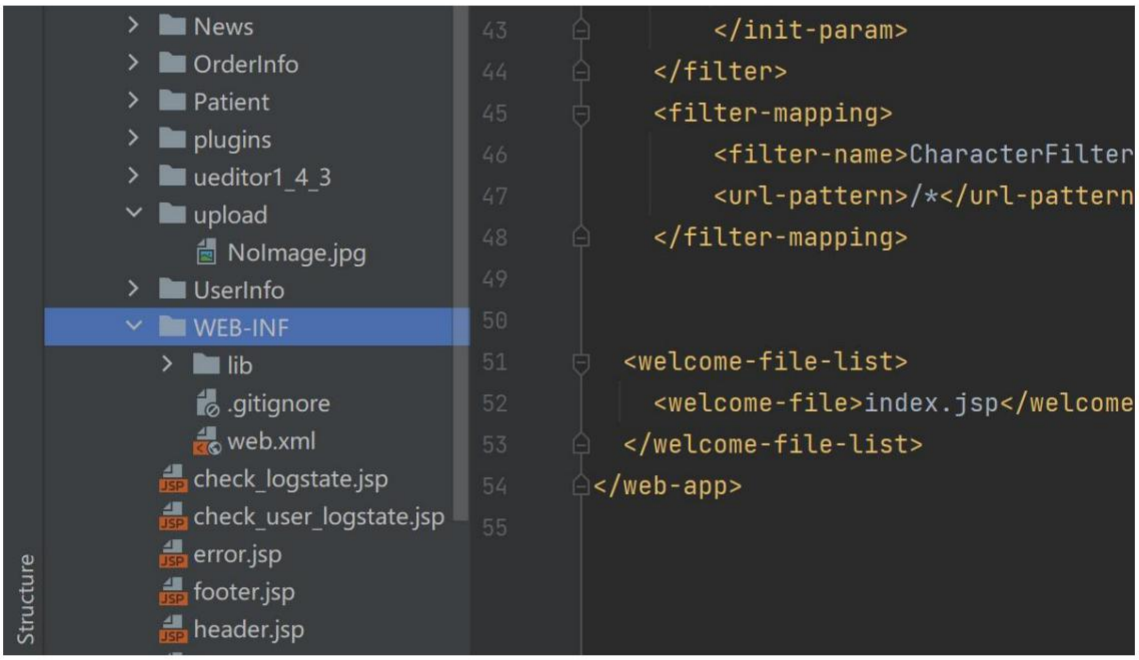
本次实验部分:

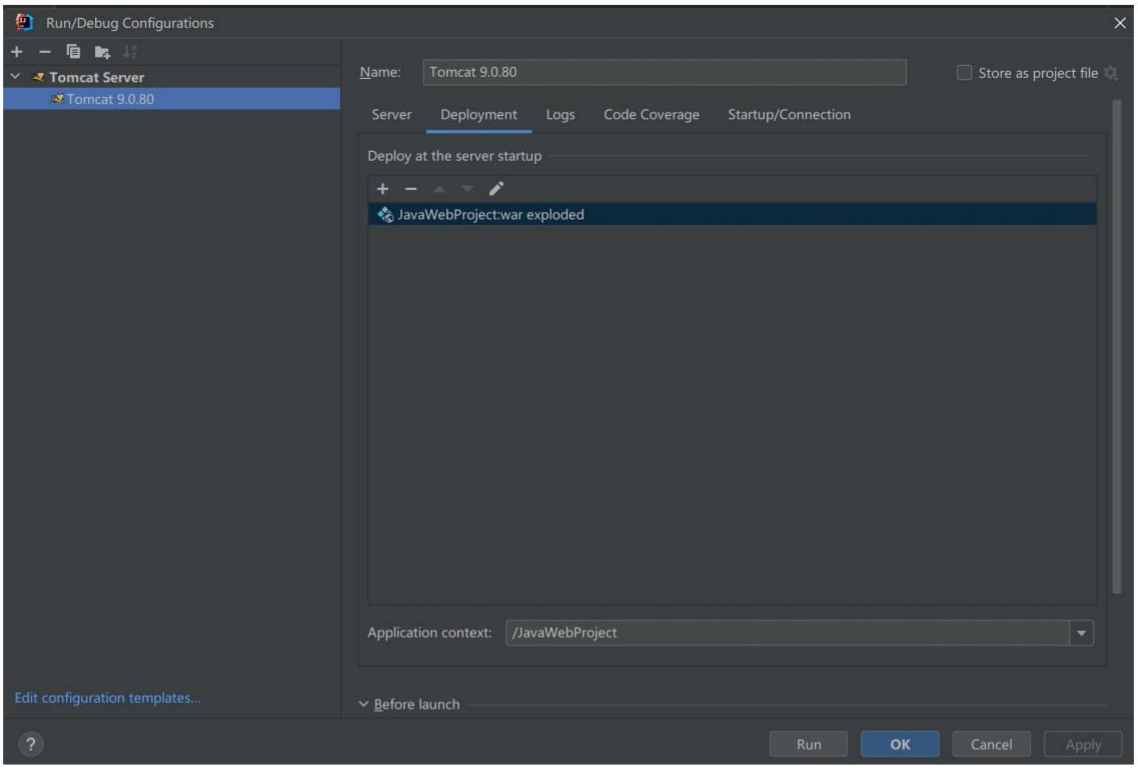
1.JavaWeb 项目的创建流程

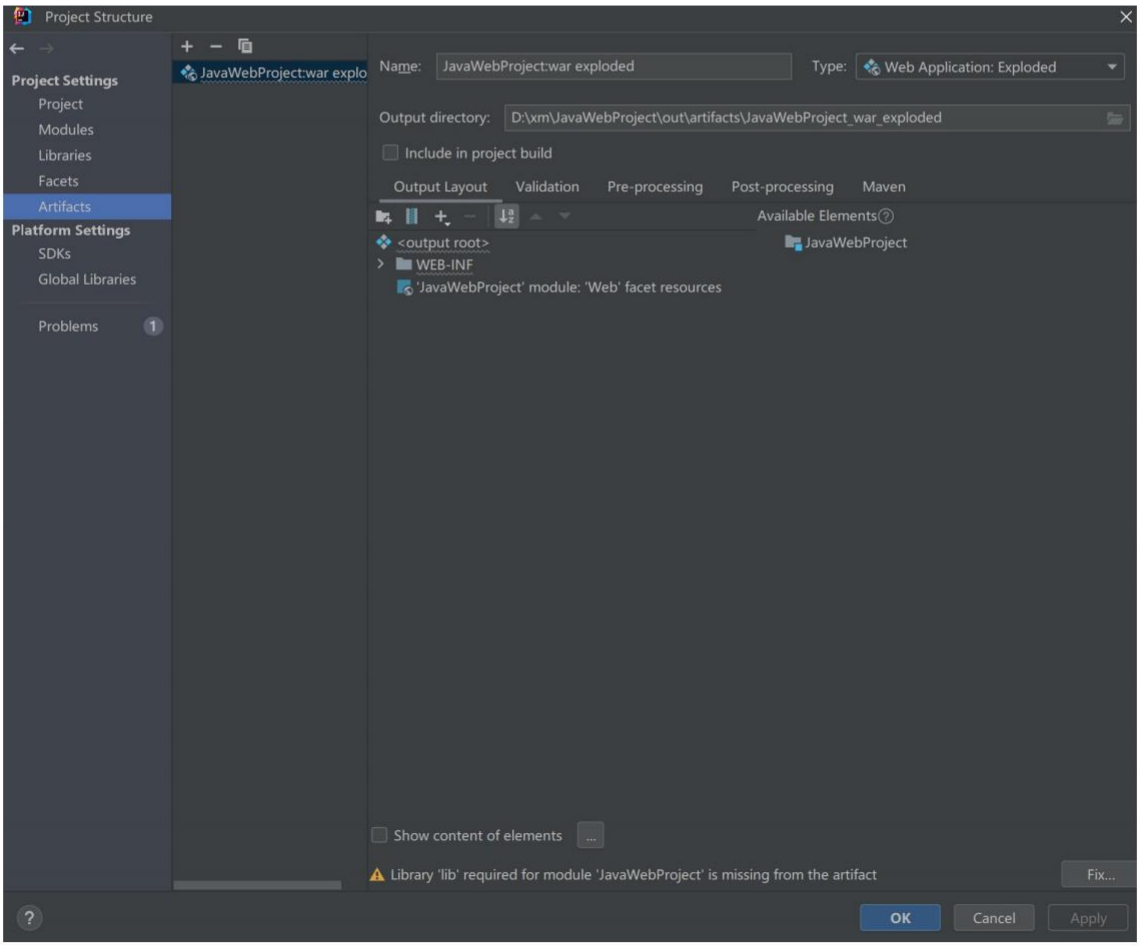
依照流程操作如下：

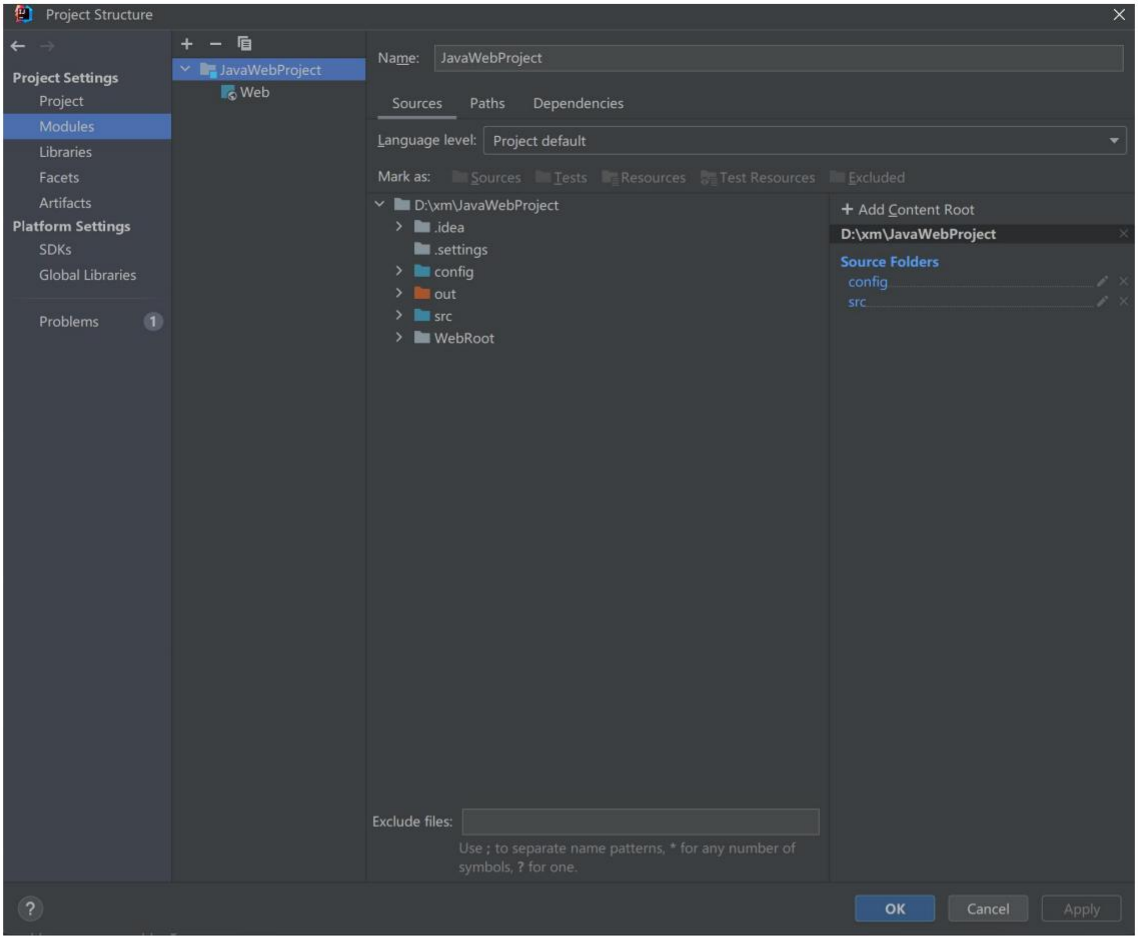
创建普通 maven 项目，add famework support 添加 web application 后得到 web 文件夹。配置 tomocat 启动，确保 modules 中项目有 web。






Run/Debug Configurations
Tomcat Server
Tomcat 9.0.80
Name: Tomcat 9.0.80
Store as project file
Server
Deployment
Logs
Code Coverage
Startup/Connection
Deploy at the server startup
JavaWebProject:war exploded
Application context: /JavaWebProject
Edit configuration templates...
Before launch
Run
OK
Cancel
Apply


Project Structure
Project Settings
Project
Modules
Libraries
Facets
Artifacts
Platform Settings
SDKs
Global Libraries
Problems
JavaWebProject:war explo
Name: JavaWebProject:war exploded
Type: Web Application: Exploded
Output directory: D:\xm\JavaWebProject\out\artifacts\JavaWebProject_war_exploded
Include in project build
Output Layout
Validation
Pre-processing
Post-processing
Maven
Available Elements
<output root>
WEB-INF
'JavaWebProject' module: 'Web' facet resources
JavaWebProject
Show content of elements
Library 'lib' required for module 'JavaWebProject' is missing from the artifact
Fix...
OK
Cancel
Apply



其中 java 代码在 src 中，网页有关在 WebRoot 中，WEB-INF 中的 web.xml 为配置文件。

2.关键核心代码展示(controller,mapper,po,srvice，utils 包，以及配置文件 web.xml)

controller 包:

BaseController:

BaseController.java

```java
package com.chengxusheji.controller;

import java.beans.PropertyEditorSupport;
import java.io.File;
import java.io.IOException;
import java.io.PrintWriter;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.UUID;

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

import org.json.JSONException;
import org.json.JSONObject;
import org.springframework.beans.propertyeditors.CustomDateEditor;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.multipart.MultipartFile;
import org.springframework.web.multipart.MultipartHttpServletRequest;

import com.chengxusheji.utils.UserException;

8 usages  8 inheritors
public class BaseController {



    @InitBinder
    // 必须有一个参数WebDataBinder
    public void initBinder(WebDataBinder binder) {
        //System.out.println(binder.getFieldDefaultPrefix());
        binder.registerCustomEditor(Date.class, new CustomDateEditor(
                new SimpleDateFormat( pattern: "yyyy-MM-dd"),  allowEmpty: false));

        binder.registerCustomEditor(Integer.class, new PropertyEditorSupport() {
            no usages
            @Override
```

```
            public String getAsText() { return (getValue() == null) ? "" : getValue().toString(); }
            no usages
            @Override
            public void setAsText(String text) {
                Integer value = null;
                if (null != text && !text.equals("")) {
                        try {
                            value = Integer.valueOf(text);
                        } catch(Exception ex)  {
                            throw new UserException("数据格式输入不正确！");
                        }
                }
                setValue(value);
            }
        });

        //binder.registerCustomEditor(Integer.class, null,new CustomNumberEditor(Integer.class, null, true));

        binder.registerCustomEditor(Float.class, new PropertyEditorSupport() {
            no usages
            @Override
            public String getAsText() { return (getValue() == null)? "" : getValue().toString(); }
            no usages
            @Override
            public void setAsText(String text)  {
                Float value = null;
                if (null != text && !text.equals("")) {
                    try {
                        value = Float.valueOf(text);
                    } catch (Exception e) {
                        throw new UserException("数据格式输入不正确！");
                    }
                }
                setValue(value);
            }
        });
    }
```

```
    /**
     * 处理图片文件上传，返回保存的文件名路径
     * fileKeyName: 图片上传表单key
     * @throws IOException
     * @throws IllegalStateException
     */
    6 usages
    public String handlePhotoUpload(HttpServletRequest request,String fileKeyName) throws IllegalStateException, IOException {
        String fileName = "upload/NoImage.jpg";
        MultipartHttpServletRequest multipartRequest = (MultipartHttpServletRequest) request;
        /**构建图片保存的目录**/
        String photoBookPathDir = "/upload";
        /**得到图片保存目录的真实路径**/
        String photoBookRealPathDir = request.getSession().getServletContext().getRealPath(photoBookPathDir);
        /**根据真实路径创建目录**/
        File photoBookSaveFile = new File(photoBookRealPathDir);
        if(!photoBookSaveFile.exists())
            photoBookSaveFile.mkdirs();
        /**页面控件的文件流**/
        MultipartFile multipartFile_photoBook = multipartRequest.getFile(fileKeyName);
        if(!multipartFile_photoBook.isEmpty()) {
            /**获取文件的后缀**/
            String suffix = multipartFile_photoBook.getOriginalFilename().substring
                            (multipartFile_photoBook.getOriginalFilename().lastIndexOf( str: "."));
            String smallSuffix = suffix.toLowerCase();
            if(!smallSuffix.equals(".jpg") && !smallSuffix.equals(".gif") && !smallSuffix.equals(".png") )
                throw new UserException("图片格式不正确！");
            /**使用UUID生成文件名称**/
            String photoBookFileName = UUID.randomUUID().toString()+ suffix;//构建文件名称
            //String logImageName = multipartFile.getOriginalFilename();
            /**拼成完整的文件保存路径加文件**/
            String photoBookFilePath = photoBookRealPathDir + File.separator  + photoBookFileName;
            File photoBookFile = new File(photoBookFilePath);

            multipartFile_photoBook.transferTo(photoBookFile);
```

```
            fileName = "upload/" + photoBookFileName;
        }

        return fileName;
    }


    /**
     * 处理图片文件上传，返回保存的文件名路径
     * fileKeyName: 图片上传表单key
     * @throws IOException
     * @throws IllegalStateException
     */
    no usages
    public String handleFileUpload(HttpServletRequest request,String fileKeyName) throws IllegalStateException, IOException {
        String fileName = "";
        MultipartHttpServletRequest multipartRequest = (MultipartHttpServletRequest) request;
        /**构建图片保存的目录**/
        String photoBookPathDir = "/upload";
        /**得到图片保存目录的真实路径**/
        String photoBookRealPathDir = request.getSession().getServletContext().getRealPath(photoBookPathDir);
        /**根据真实路径创建目录**/
        File photoBookSaveFile = new File(photoBookRealPathDir);
        if(!photoBookSaveFile.exists())
            photoBookSaveFile.mkdirs();
        /**页面控件的文件流**/
        MultipartFile multipartFile_photoBook = multipartRequest.getFile(fileKeyName);
        if(!multipartFile_photoBook.isEmpty()) {
            /**获取文件的后缀**/
            String suffix = multipartFile_photoBook.getOriginalFilename().substring
                            (multipartFile_photoBook.getOriginalFilename().lastIndexOf( str: "."));
            /**使用UUID生成文件名称**/
            String photoBookFileName = UUID.randomUUID().toString()+ suffix;//构建文件名称
            //String logImageName = multipartFile.getOriginalFilename();
            /**拼成完整的文件保存路径加文件**/
            String photoBookFilePath = photoBookRealPathDir + File.separator  + photoBookFileName;
```

```
            File photoBookFile = new File(photoBookFilePath);

            multipartFile_photoBook.transferTo(photoBookFile);

            fileName = "upload/" + photoBookFileName;
        }

        return fileName;
    }


    /*向客户端输出操作成功或失败信息*/
    59 usages
    public void writeJsonResponse(HttpServletResponse response,boolean success,String message)
            throws IOException, JSONException {
        response.setContentType("text/json;charset=UTF-8");
        PrintWriter out = response.getWriter();
        //将要被返回到客户端的对象
        JSONObject json=new JSONObject();
        json.accumulate("success", success);
        json.accumulate("message", message);
        out.println(json.toString());
        out.flush();
        out.close();
    }


}
```

DepartmentController:

DepartmentController.java

```java
package com.chengxusheji.controller;

import java.io.IOException;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.io.UnsupportedEncodingException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.List;

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

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import com.chengxusheji.utils.ExportExcelUtil;
import com.chengxusheji.utils.UserException;
import com.chengxusheji.service.DepartmentService;
import com.chengxusheji.po.Department;

//Department管理控制层
@Controller
@RequestMapping("/Department")
public class DepartmentController extends BaseController {

    /*业务层对象*/
    17 usages
    @Resource DepartmentService departmentService;

    @InitBinder("department")
    public void initBinderDepartment(WebDataBinder binder) { binder.setFieldDefaultPrefix("department."); }
    /*跳转到添加Department视图*/
    @RequestMapping(value = "/add", method = RequestMethod.GET)
    public String add(Model model,HttpServletRequest request) throws Exception {
        model.addAttribute(new Department());
        return "Department_add";
    }

    /*客户端ajax方式提交添加科室信息信息*/
    @RequestMapping(value = "/add", method = RequestMethod.POST)
    public void add(@Validated Department department, BindingResult br,
            Model model, HttpServletRequest request,HttpServletResponse response) throws Exception {
        String message = "";
        boolean success = false;
        if (br.hasErrors()) {
            message = "输入信息不符合要求！";
            writeJsonResponse(response, success, message);
            return ;
        }
        departmentService.addDepartment(department);
        message = "科室信息添加成功!";
        success = true;
        writeJsonResponse(response, success, message);
    }
    /*ajax方式按照查询条件分页查询科室信息信息*/
    @RequestMapping(value = { "/list" }, method = {RequestMethod.GET,RequestMethod.POST})
    public void list(String departmentName,String birthDate,Integer page,Integer rows, Model model, HttpServletRequest request,HttpServletResp
        if (page==null || page == 0) page = 1;
```

```
        if (departmentName == null) departmentName = "";
        if (birthDate == null) birthDate = "";
        if(rows != 0)departmentService.setRows(rows);
        List<Department> departmentList = departmentService.queryDepartment(departmentName, birthDate, page);
        /*计算总的页数和总的记录数*/
        departmentService.queryTotalPageAndRecordNumber(departmentName, birthDate);
        /*获取到总的页码数目*/
        int totalPage = departmentService.getTotalPage();
        /*当前查询条件下总记录数*/
        int recordNumber = departmentService.getRecordNumber();
        response.setContentType("text/json;charset=UTF-8");
        PrintWriter out = response.getWriter();
        //将要被返回到客户端的对象
        JSONObject jsonObj=new JSONObject();
        jsonObj.accumulate("total", recordNumber);
        JSONArray jsonArray = new JSONArray();
        for(Department department:departmentList) {
            JSONObject jsonDepartment = department.getJsonObject();
            jsonArray.put(jsonDepartment);
        }
        jsonObj.accumulate("rows", jsonArray);
        out.println(jsonObj.toString());
        out.flush();
        out.close();
    }

    /*ajax方式按照查询条件分页查询科室信息信息*/
    @RequestMapping(value = { "/listAll" }, method = {RequestMethod.GET,RequestMethod.POST})
    public void listAll(HttpServletResponse response) throws Exception {
        List<Department> departmentList = departmentService.queryAllDepartment();
        response.setContentType("text/json;charset=UTF-8");
        PrintWriter out = response.getWriter();
        JSONArray jsonArray = new JSONArray();
        for(Department department:departmentList) {
            JSONObject jsonDepartment = new JSONObject();
```

```
            jsonDepartment.accumulate("departmentId", department.getDepartmentId());
            jsonDepartment.accumulate("departmentName", department.getDepartmentName());
            jsonArray.put(jsonDepartment);
        }
        out.println(jsonArray.toString());
        out.flush();
        out.close();
    }

    /*前台按照查询条件分页查询科室信息信息*/
    @RequestMapping(value = { "/frontlist" }, method = {RequestMethod.GET,RequestMethod.POST})
    public String frontlist(String departmentName,String birthDate,Integer currentPage, Model model, HttpServletRequest request) throws Except
        if (currentPage==null || currentPage == 0) currentPage = 1;
        if (departmentName == null) departmentName = "";
        if (birthDate == null) birthDate = "";
        List<Department> departmentList = departmentService.queryDepartment(departmentName, birthDate, currentPage);
        /*计算总的页数和总的记录数*/
        departmentService.queryTotalPageAndRecordNumber(departmentName, birthDate);
        /*获取到总的页码数目*/
        int totalPage = departmentService.getTotalPage();
        /*当前查询条件下总记录数*/
        int recordNumber = departmentService.getRecordNumber();
        request.setAttribute( s: "departmentList",  departmentList);
        request.setAttribute( s: "totalPage", totalPage);
        request.setAttribute( s: "recordNumber", recordNumber);
        request.setAttribute( s: "currentPage", currentPage);
        request.setAttribute( s: "departmentName", departmentName);
        request.setAttribute( s: "birthDate", birthDate);
        return "Department/department_frontquery_result";
    }

     /*前台查询Department信息*/
    @RequestMapping(value="/{departmentId}/frontshow",method=RequestMethod.GET)
    public String frontshow(@PathVariable Integer departmentId,Model model,HttpServletRequest request) throws Exception {
        /*根据主键departmentId获取Department对象*/
        Department department = departmentService.getDepartment(departmentId);
```

```
        request.setAttribute( s: "department",  department);
        return "Department/department_frontshow";
    }

    /*ajax方式显示科室信息修改jsp视图页*/
    @RequestMapping(value="/{departmentId}/update",method=RequestMethod.GET)
    public void update(@PathVariable Integer departmentId,Model model,HttpServletRequest request,HttpServletResponse response) throws Exceptio
        /*根据主键departmentId获取Department对象*/
        Department department = departmentService.getDepartment(departmentId);

        response.setContentType("text/json;charset=UTF-8");
        PrintWriter out = response.getWriter();
        //将要被返回到客户端的对象
        JSONObject jsonDepartment = department.getJsonObject();
        out.println(jsonDepartment.toString());
        out.flush();
        out.close();
    }

    /*ajax方式更新科室信息信息*/
    @RequestMapping(value = "/{departmentId}/update", method = RequestMethod.POST)
    public void update(@Validated Department department, BindingResult br,
            Model model, HttpServletRequest request,HttpServletResponse response) throws Exception {
        String message = "";
        boolean success = false;
        if (br.hasErrors()) {
            message = "输入的信息有错误！";
            writeJsonResponse(response, success, message);
            return;
        }
        try {
            departmentService.updateDepartment(department);
            message = "科室信息更新成功!";
            success = true;
            writeJsonResponse(response, success, message);
        } catch (Exception e) {
            writeJsonResponse(response, success, message);
        } catch (Exception e) {
            e.printStackTrace();
            message = "科室信息更新失败!";
            writeJsonResponse(response, success, message);
        }
    }
    /*删除科室信息信息*/
    @RequestMapping(value="/{departmentId}/delete",method=RequestMethod.GET)
    public String delete(@PathVariable Integer departmentId,HttpServletRequest request) throws UnsupportedEncodingException {
          try {
              departmentService.deleteDepartment(departmentId);
                request.setAttribute( s: "message",  o: "科室信息删除成功!");
                return "message";
            } catch (Exception e) {
                e.printStackTrace();
                request.setAttribute( s: "error",  o: "科室信息删除失败!");
                return "error";

            }

    }

    /*ajax方式删除多条科室信息记录*/
    @RequestMapping(value="/deletes",method=RequestMethod.POST)
    public void delete(String departmentIds,HttpServletRequest request,HttpServletResponse response) throws IOException, JSONException {
        String message = "";
        boolean success = false;
        try {
            int count = departmentService.deleteDepartments(departmentIds);
            success = true;
            message = count + "条记录删除成功";
            writeJsonResponse(response, success, message);
        } catch (Exception e) {
            //e.printStackTrace();
            message = "有记录存在外键约束,删除失败";
            writeJsonResponse(response, success, message);
        }
```

```
}

/*按照查询条件导出科室信息信息到Excel*/
@RequestMapping(value = { "/OutToExcel" }, method = {RequestMethod.GET,RequestMethod.POST})
public void OutToExcel(String departmentName,String birthDate, Model model, HttpServletRequest request,HttpServletResponse response) th
    if(departmentName == null) departmentName = "";
    if(birthDate == null) birthDate = "";
    List<Department> departmentList = departmentService.queryDepartment(departmentName,birthDate);
    ExportExcelUtil ex = new ExportExcelUtil();
    String _title = "Department信息记录";
    String[] headers = { "科室id","科室名称","成立日期","负责人"};
    List<String[]> dataset = new ArrayList<>();
    for(int i=0;i<departmentList.size();i++) {
        Department department = departmentList.get(i);
        dataset.add(new String[]{department.getDepartmentId() + "",department.getDepartmentName(),department.getBirthDate(),department.
    }
    /*
    OutputStream out = null;
    try {
        out = new FileOutputStream("C://output.xls");
        ex.exportExcel(title,headers, dataset, out);
        out.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
    */
    OutputStream out = null;//创建一个输出流对象
```

```
    try {
        out = response.getOutputStream();//
        response.setHeader( s: "Content-disposition", s1: "attachment; filename="+"Department.xls");//filename是下载的xls的名，建议最好用英文
        response.setContentType("application/msexcel;charset=UTF-8");//设置类型
        response.setHeader( s: "Pragma", s1: "No-cache");//设置头
        response.setHeader( s: "Cache-Control", s1: "no-cache");//设置头
        response.setDateHeader( s: "Expires",  l: 0);//设置日期头
        String rootPath = request.getSession().getServletContext().getRealPath( s: "/");
        ex.exportExcel(rootPath,_title,headers, dataset, out);
        out.flush();
    } catch (IOException e) {
        e.printStackTrace();
    }finally{
        try{
            if(out!=null){
                out.close();
            }
        }catch(IOException e){
            e.printStackTrace();
        }
    }
}
```

DoctorController:

DoctorController.java

```java
package com.chengxusheji.controller;

import com.chengxusheji.po.Admin;
import com.chengxusheji.service.AdminService;
import java.io.IOException;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.io.UnsupportedEncodingException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.List;

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

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import com.chengxusheji.utils.ExportExcelUtil;
import com.chengxusheji.utils.UserException;
import com.chengxusheji.service.DoctorService;
import com.chengxusheji.po.Doctor;
import com.chengxusheji.service.DepartmentService;
import com.chengxusheji.po.Department;

```

```java
//Doctor管理控制层
@Controller
@RequestMapping("/Doctor")
public class DoctorController extends BaseController {

    /*业务层对象*/
    18 usages
    @Resource DoctorService doctorService;

    3 usages
    @Resource DepartmentService departmentService;

    3 usages
    @Resource
    AdminService adminService;
    @InitBinder("departmentObj")
    public void initBinderdepartmentObj(WebDataBinder binder) { binder.setFieldDefaultPrefix("departmentObj."); }
    @InitBinder("doctor")
    public void initBinderDoctor(WebDataBinder binder) { binder.setFieldDefaultPrefix("doctor."); }
    /*跳转到添加Doctor视图*/
    @RequestMapping(value = "/add", method = RequestMethod.GET)
    public String add(Model model,HttpServletRequest request) throws Exception {
        model.addAttribute(new Doctor());
        /*查询所有的Department信息*/
        List<Department> departmentList = departmentService.queryAllDepartment();
        request.setAttribute( s: "departmentList", departmentList);
        return "Doctor_add";
    }

    /*客户端ajax方式提交添加医生信息信息*/
    @RequestMapping(value = "/add", method = RequestMethod.POST)
    public void add(@Validated Doctor doctor, BindingResult br,
            Model model, HttpServletRequest request,HttpServletResponse response) throws Exception {
        String message = "";
        boolean success = false;
        if (br.hasErrors()) {
            message = "输入信息不符合要求！";
            writeJsonResponse(response, success, message);
            return ;
        }
        if(doctorService.getDoctor(doctor.getDoctorNumber()) != null) {
            message = "医生工号已经存在！";
            writeJsonResponse(response, success, message);
            return ;
        }
        Admin admin =adminService.findAdminByUserName(doctor.getDoctorNumber());
        if(admin != null) {
            message = "用户名已经存在！";
            writeJsonResponse(response, success, message);
            return ;
        }
        try {
            doctor.setDoctorPhoto(this.handlePhotoUpload(request,  fileKeyName: "doctorPhotoFile"));
        } catch(UserException ex) {
            message = "图片格式不正确！";
            writeJsonResponse(response, success, message);
            return ;
        }
            admin = new Admin();
            admin.setPassword(doctor.getPassword());
            admin.setUsername(doctor.getDoctorNumber());
            admin.setType(1);
            adminService.addAdmin(admin);
            admin = adminService.findAdminByUserName(doctor.getDoctorNumber());
            doctor.setAdminid(admin.getId());
            doctorService.addDoctor(doctor);
        message = "医生信息添加成功!";
        success = true;
        writeJsonResponse(response, success, message);
    }
    /*ajax方式按照查询条件分页查询医生信息信息*/
```

```
    @RequestMapping(value = { "/list" }, method = {RequestMethod.GET,RequestMethod.POST})
    public void list(String doctorNumber,@ModelAttribute("departmentObj") Department departmentObj,String doctorName,String birthDate,Integ
        if (page==null || page == 0) page = 1;
        if (doctorNumber == null) doctorNumber = "";
        if (doctorName == null) doctorName = "";
        if (birthDate == null) birthDate = "";
        if(rows != 0)doctorService.setRows(rows);
        List<Doctor> doctorList = doctorService.queryDoctor(doctorNumber, departmentObj, doctorName, birthDate, page);
        /*计算总的页数和总的记录数*/
        doctorService.queryTotalPageAndRecordNumber(doctorNumber, departmentObj, doctorName, birthDate);
        /*获取到总的页码数目*/
        int totalPage = doctorService.getTotalPage();
        /*当前查询条件下总记录数*/
        int recordNumber = doctorService.getRecordNumber();
        response.setContentType("text/json;charset=UTF-8");
        PrintWriter out = response.getWriter();
        //将要被返回到客户端的对象
        JSONObject jsonObj=new JSONObject();
        jsonObj.accumulate("total", recordNumber);
        JSONArray jsonArray = new JSONArray();
        for(Doctor doctor:doctorList) {
            JSONObject jsonDoctor = doctor.getJsonObject();
            jsonArray.put(jsonDoctor);
        }
        jsonObj.accumulate("rows", jsonArray);
        out.println(jsonObj.toString());
        out.flush();
        out.close();
    }

    /*ajax方式按照查询条件分页查询医生信息信息*/
    @RequestMapping(value = { "/listAll" }, method = {RequestMethod.GET,RequestMethod.POST})
    public void listAll(HttpServletResponse response) throws Exception {
        List<Doctor> doctorList = doctorService.queryAllDoctor();
        response.setContentType("text/json;charset=UTF-8");
        PrintWriter out = response.getWriter();
        JSONArray jsonArray = new JSONArray();
        for(Doctor doctor:doctorList) {
            JSONObject jsonDoctor = new JSONObject();
            jsonDoctor.accumulate("doctorNumber", doctor.getDoctorNumber());
            jsonDoctor.accumulate("doctorName", doctor.getDoctorName());
            jsonArray.put(jsonDoctor);
        }
        out.println(jsonArray.toString());
        out.flush();
        out.close();
    }

    /*前台按照查询条件分页查询医生信息信息*/
    @RequestMapping(value = { "/frontlist" }, method = {RequestMethod.GET,RequestMethod.POST})
    public String frontlist(String doctorNumber,@ModelAttribute("departmentObj") Department departmentObj,String doctorName,String birthDate,
        if (currentPage==null || currentPage == 0) currentPage = 1;
        if (doctorNumber == null) doctorNumber = "";
        if (doctorName == null) doctorName = "";
        if (birthDate == null) birthDate = "";
        List<Doctor> doctorList = doctorService.queryDoctor(doctorNumber, departmentObj, doctorName, birthDate, currentPage);
        /*计算总的页数和总的记录数*/
        doctorService.queryTotalPageAndRecordNumber(doctorNumber, departmentObj, doctorName, birthDate);
        /*获取到总的页码数目*/
        int totalPage = doctorService.getTotalPage();
        /*当前查询条件下总记录数*/
        int recordNumber = doctorService.getRecordNumber();
        request.setAttribute( s: "doctorList",  doctorList);
        request.setAttribute( s: "totalPage", totalPage);
        request.setAttribute( s: "recordNumber", recordNumber);
        request.setAttribute( s: "currentPage", currentPage);
        request.setAttribute( s: "doctorNumber", doctorNumber);
        request.setAttribute( s: "departmentObj", departmentObj);
        request.setAttribute( s: "doctorName", doctorName);
        request.setAttribute( s: "birthDate", birthDate);
        List<Department> departmentList = departmentService.queryAllDepartment();
        request.setAttribute( s: "departmentList", departmentList);
        return "Doctor/doctor_frontquery_result";
```

```
    }

     /*前台查询Doctor信息*/
    @RequestMapping(value="/{doctorNumber}/frontshow",method=RequestMethod.GET)
    public String frontshow(@PathVariable String doctorNumber,Model model,HttpServletRequest request) throws Exception {
        /*根据主键doctorNumber获取Doctor对象*/
        Doctor doctor = doctorService.getDoctor(doctorNumber);

        List<Department> departmentList = departmentService.queryAllDepartment();
        request.setAttribute( s: "departmentList", departmentList);
        request.setAttribute( s: "doctor",  doctor);
        return "Doctor/doctor_frontshow";
    }

    /*ajax方式显示医生信息修改jsp视图页*/
    @RequestMapping(value="/{doctorNumber}/update",method=RequestMethod.GET)
    public void update(@PathVariable String doctorNumber,Model model,HttpServletRequest request,HttpServletResponse response) throws Exceptio
        /*根据主键doctorNumber获取Doctor对象*/
        Doctor doctor = doctorService.getDoctor(doctorNumber);

        response.setContentType("text/json;charset=UTF-8");
        PrintWriter out = response.getWriter();
        //将要被返回到客户端的对象
        JSONObject jsonDoctor = doctor.getJsonObject();
        out.println(jsonDoctor.toString());
        out.flush();
        out.close();
    }

    /*ajax方式更新医生信息信息*/
    @RequestMapping(value = "/{doctorNumber}/update", method = RequestMethod.POST)
    public void update(@Validated Doctor doctor, BindingResult br,
            Model model, HttpServletRequest request,HttpServletResponse response) throws Exception {
        String message = "";
        boolean success = false;
        if (br.hasErrors()) {
            message = "输入的信息有错误！";
            writeJsonResponse(response, success, message);
            return;
        }
        String doctorPhotoFileName = this.handlePhotoUpload(request,  fileKeyName: "doctorPhotoFile");
        if(!doctorPhotoFileName.equals("upload/NoImage.jpg"))doctor.setDoctorPhoto(doctorPhotoFileName);


        try {
            doctorService.updateDoctor(doctor);
            message = "医生信息更新成功!";
            success = true;
            writeJsonResponse(response, success, message);
        } catch (Exception e) {
            e.printStackTrace();
            message = "医生信息更新失败!";
            writeJsonResponse(response, success, message);
        }
    }
    /*删除医生信息信息*/
    @RequestMapping(value="/{doctorNumber}/delete",method=RequestMethod.GET)
    public String delete(@PathVariable String doctorNumber,HttpServletRequest request) throws UnsupportedEncodingExcepti
          try {
              doctorService.deleteDoctor(doctorNumber);
                request.setAttribute( s: "message",  o: "医生信息删除成功!");
                return "message";
            } catch (Exception e) {
                e.printStackTrace();
                request.setAttribute( s: "error",  o: "医生信息删除失败!");
                return "error";

            }

    }
```

```java
    /*ajax方式删除多条医生信息记录*/
    @RequestMapping(value="/deletes",method=RequestMethod.POST)
    public void delete(String doctorNumbers,HttpServletRequest request,HttpServletResponse response) throws IOException, JSONException {
        String message = "";
        boolean success = false;
        try {
            int count = doctorService.deleteDoctors(doctorNumbers);
            success = true;
            message = count + "条记录删除成功";
            writeJsonResponse(response, success, message);
        } catch (Exception e) {
            //e.printStackTrace();
            message = "有记录存在外键约束,删除失败";
            writeJsonResponse(response, success, message);
        }
    }

    /*按照查询条件导出医生信息信息到Excel*/
    @RequestMapping(value = { "/OutToExcel" }, method = {RequestMethod.GET,RequestMethod.POST})
    public void OutToExcel(String doctorNumber,@ModelAttribute("departmentObj") Department departmentObj,String doctorName,String birthDate,
        if(doctorNumber == null) doctorNumber = "";
        if(doctorName == null) doctorName = "";
        if(birthDate == null) birthDate = "";
        List<Doctor> doctorList = doctorService.queryDoctor(doctorNumber,departmentObj,doctorName,birthDate);
        ExportExcelUtil ex = new ExportExcelUtil();
        String _title = "Doctor信息记录";
        String[] headers = { "医生工号","所在科室","医生姓名","性别","医生照片","出生日期","医生职位","工作经验","联系方式"};
        List<String[]> dataset = new ArrayList<>();
        for(int i=0;i<doctorList.size();i++) {
            Doctor doctor = doctorList.get(i);
            dataset.add(new String[]{doctor.getDoctorNumber(),doctor.getDepartmentObj().getDepartmentName(),doctor.getDoctorName(),doctor.ge
        }
        /*
        OutputStream out = null;
        try {
            out = new FileOutputStream("C://output.xls");
            ex.exportExcel(title,headers, dataset, out);
            out.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
        */
        OutputStream out = null;//创建一个输出流对象
        try {
            out = response.getOutputStream();//
            response.setHeader( s: "Content-disposition", s1: "attachment; filename="+"Doctor.xls");//filename是下载的xls的名，建议最好用英文
            response.setContentType("application/msexcel;charset=UTF-8");//设置类型
            response.setHeader( s: "Pragma", s1: "No-cache");//设置头
            response.setHeader( s: "Cache-Control", s1: "no-cache");//设置头
            response.setDateHeader( s: "Expires",  l: 0);//设置日期头
            String rootPath = request.getSession().getServletContext().getRealPath( s: "/");
            ex.exportExcel(rootPath,_title,headers, dataset, out);
            out.flush();
        } catch (IOException e) {
            e.printStackTrace();
        }finally{
            try{
                if(out!=null){
                    out.close();
                }
            }catch(IOException e){
                e.printStackTrace();
            }
        }
    }
}
```

LeavewordController：

LeavewordController.java

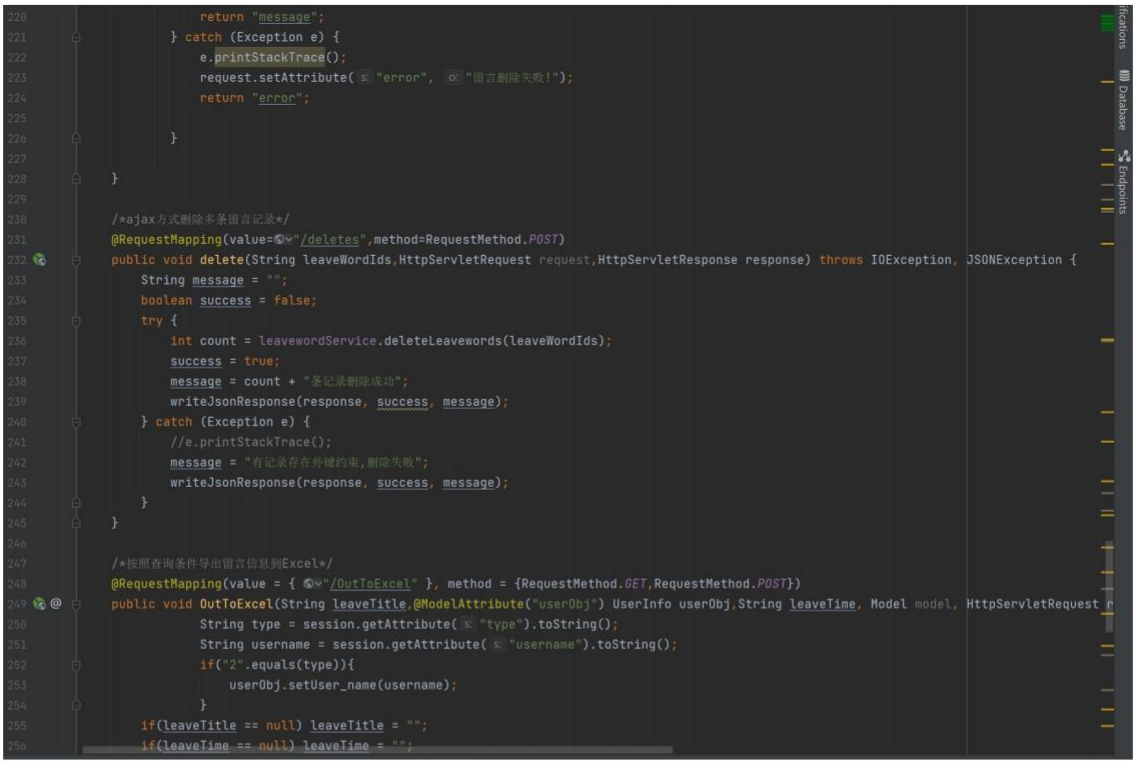
```java
package com.chengxusheji.controller;

import java.io.IOException;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.io.UnsupportedEncodingException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.List;

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

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import com.chengxusheji.utils.ExportExcelUtil;
import com.chengxusheji.utils.UserException;
import com.chengxusheji.service.LeavewordService;
import com.chengxusheji.po.Leaveword;
import com.chengxusheji.service.UserInfoService;
import com.chengxusheji.po.UserInfo;

//Leaveword管理控制层
@Controller
@RequestMapping("/Leaveword")
public class LeavewordController extends BaseController {

    /*业务层对象*/
    17 usages
    @Resource LeavewordService leavewordService;

    3 usages
    @Resource UserInfoService userInfoService;
    @InitBinder("userObj")
    public void initBinderuserObj(WebDataBinder binder) { binder.setFieldDefaultPrefix("userObj."); }
    @InitBinder("leaveword")
    public void initBinderLeaveword(WebDataBinder binder) { binder.setFieldDefaultPrefix("leaveword."); }
    /*跳转到添加Leaveword视图*/
    @RequestMapping(value = "/add", method = RequestMethod.GET)
    public String add(Model model,HttpServletRequest request) throws Exception {
        model.addAttribute(new Leaveword());
        /*查询所有的UserInfo信息*/
        List<UserInfo> userInfoList = userInfoService.queryAllUserInfo();
        request.setAttribute( s: "userInfoList", userInfoList);
        return "Leaveword_add";
    }

    /*客户端ajax方式提交添加留言信息*/
    @RequestMapping(value = "/add", method = RequestMethod.POST)
    public void add(@Validated Leaveword leaveword, BindingResult br,
            Model model, HttpServletRequest request,HttpServletResponse response,HttpSession session) throws Exception {
        String message = "";
        boolean success = false;
        UserInfo userInfo = new UserInfo();
        userInfo.setUser_name(session.getAttribute( s: "username").toString());
        leaveword.setUserObj(userInfo);
        if (br.hasErrors()) {
            message = "输入信息不符合要求！";
            writeJsonResponse(response, success, message);
            return ;
        }
```

```
        leavewordService.addLeaveword(leaveword);
        message = "留言添加成功!";
        success = true;
        writeJsonResponse(response, success, message);
    }
    /*ajax方式按照查询条件分页查询留言信息*/
    @RequestMapping(value = { "/list" }, method = {RequestMethod.GET,RequestMethod.POST})
    public void list(String leaveTitle,@ModelAttribute("userObj") UserInfo userObj,String leaveTime,Integer page,Integer rows, Model model
        String type = session.getAttribute( s: "type").toString();
        String username = session.getAttribute( s: "username").toString();
        if("2".equals(type)){
            userObj.setUser_name(username);
        }
        if (page==null || page == 0) page = 1;
        if (leaveTitle == null) leaveTitle = "";
        if (leaveTime == null) leaveTime = "";
        if(rows != 0)leavewordService.setRows(rows);
        List<Leaveword> leavewordList = leavewordService.queryLeaveword(leaveTitle, userObj, leaveTime, page);
        /*计算总的页数和总的记录数*/
        leavewordService.queryTotalPageAndRecordNumber(leaveTitle, userObj, leaveTime);
        /*获取到总的页码数目*/
        int totalPage = leavewordService.getTotalPage();
        /*当前查询条件下总记录数*/
        int recordNumber = leavewordService.getRecordNumber();
        response.setContentType("text/json;charset=UTF-8");
        PrintWriter out = response.getWriter();
        //将要被返回到客户端的对象
        JSONObject jsonObj=new JSONObject();
        jsonObj.accumulate("total", recordNumber);
        JSONArray jsonArray = new JSONArray();
        for(Leaveword leaveword:leavewordList) {
            JSONObject jsonLeaveword = leaveword.getJsonObject();
            jsonArray.put(jsonLeaveword);
        }
        jsonObj.accumulate("rows", jsonArray);
```

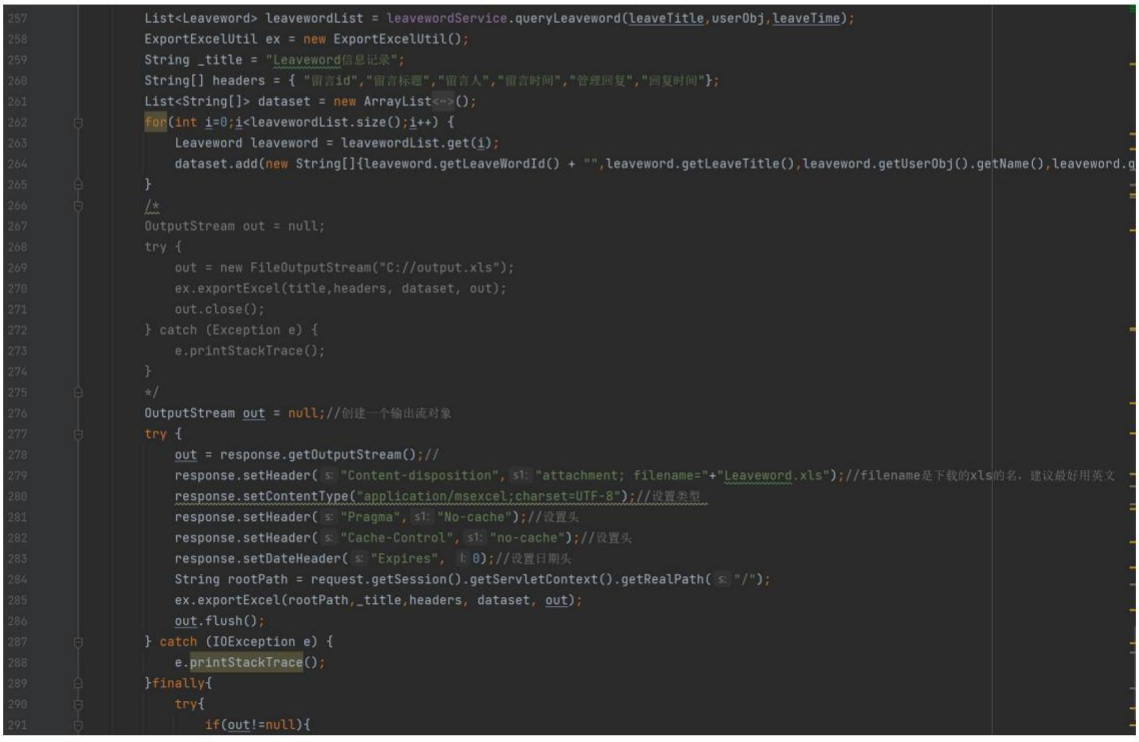
```
        out.println(jsonObj.toString());
        out.flush();
        out.close();
    }

    /*ajax方式按照查询条件分页查询留言信息*/
    @RequestMapping(value = { "/listAll" }, method = {RequestMethod.GET,RequestMethod.POST})
    public void listAll(HttpServletResponse response) throws Exception {
        List<Leaveword> leavewordList = leavewordService.queryAllLeaveword();
        response.setContentType("text/json;charset=UTF-8");
        PrintWriter out = response.getWriter();
        JSONArray jsonArray = new JSONArray();
        for(Leaveword leaveword:leavewordList) {
            JSONObject jsonLeaveword = new JSONObject();
            jsonLeaveword.accumulate("leaveWordId", leaveword.getLeaveWordId());
            jsonLeaveword.accumulate("leaveTitle", leaveword.getLeaveTitle());
            jsonArray.put(jsonLeaveword);
        }
        out.println(jsonArray.toString());
        out.flush();
        out.close();
    }

    /*前台按照查询条件分页查询留言信息*/
    @RequestMapping(value = { "/frontlist" }, method = {RequestMethod.GET,RequestMethod.POST})
    public String frontlist(String leaveTitle,@ModelAttribute("userObj") UserInfo userObj,String leaveTime,Integer currentPage, Model m
        String type = session.getAttribute( s: "type").toString();
        String username = session.getAttribute( s: "username").toString();
        if("2".equals(type)){
            userObj.setUser_name(username);
        }
        if (currentPage==null || currentPage == 0) currentPage = 1;
        if (leaveTitle == null) leaveTitle = "";
        if (leaveTime == null) leaveTime = "";
        List<Leaveword> leavewordList = leavewordService.queryLeaveword(leaveTitle, userObj, leaveTime, currentPage);
        /*计算总的页数和总的记录数*/
```

```
        leavewordService.queryTotalPageAndRecordNumber(leaveTitle, userObj, leaveTime);
        /*获取到总的页码数目*/
        int totalPage = leavewordService.getTotalPage();
        /*当前查询条件下总记录数*/
        int recordNumber = leavewordService.getRecordNumber();
        request.setAttribute( s: "leavewordList",  leavewordList);
        request.setAttribute( s: "totalPage", totalPage);
        request.setAttribute( s: "recordNumber", recordNumber);
        request.setAttribute( s: "currentPage", currentPage);
        request.setAttribute( s: "leaveTitle", leaveTitle);
        request.setAttribute( s: "userObj", userObj);
        request.setAttribute( s: "leaveTime", leaveTime);
        List<UserInfo> userInfoList = userInfoService.queryAllUserInfo();
        request.setAttribute( s: "userInfoList", userInfoList);
        return "Leaveword/leaveword_frontquery_result";
    }

     /*前台查询Leaveword信息*/
    @RequestMapping(value="/{leaveWordId}/frontshow",method=RequestMethod.GET)
    public String frontshow(@PathVariable Integer leaveWordId,Model model,HttpServletRequest request) throws Exception {
        /*根据主键leaveWordId获取Leaveword对象*/
        Leaveword leaveword = leavewordService.getLeaveword(leaveWordId);

        List<UserInfo> userInfoList = userInfoService.queryAllUserInfo();
        request.setAttribute( s: "userInfoList", userInfoList);
        request.setAttribute( s: "leaveword",  leaveword);
        return "Leaveword/leaveword_frontshow";
    }

    /*ajax方式显示留言修改jsp视图页*/
    @RequestMapping(value="/{leaveWordId}/update",method=RequestMethod.GET)
    public void update(@PathVariable Integer leaveWordId,Model model,HttpServletRequest request,HttpServletResponse response) throws Except
        /*根据主键leaveWordId获取Leaveword对象*/
        Leaveword leaveword = leavewordService.getLeaveword(leaveWordId);

        response.setContentType("text/json;charset=UTF-8");
        PrintWriter out = response.getWriter();
```

```
        //将要被返回到客户端的对象
        JSONObject jsonLeaveword = leaveword.getJsonObject();
        out.println(jsonLeaveword.toString());
        out.flush();
        out.close();
    }

    /*ajax方式更新留言信息*/
    @RequestMapping(value = "/{leaveWordId}/update", method = RequestMethod.POST)
    public void update(@Validated Leaveword leaveword, BindingResult br,
            Model model, HttpServletRequest request,HttpServletResponse response) throws Exception {
        String message = "";
        boolean success = false;
        if (br.hasErrors()) {
            message = "输入的信息有错误！";
            writeJsonResponse(response, success, message);
            return;
        }
        try {
            leavewordService.updateLeaveword(leaveword);
            message = "留言更新成功!";
            success = true;
            writeJsonResponse(response, success, message);
        } catch (Exception e) {
            e.printStackTrace();
            message = "留言更新失败!";
            writeJsonResponse(response, success, message);
        }
    }
    /*删除留言信息*/
    @RequestMapping(value="/{leaveWordId}/delete",method=RequestMethod.GET)
    public String delete(@PathVariable Integer leaveWordId,HttpServletRequest request) throws UnsupportedEncodingException
        try {
            leavewordService.deleteLeaveword(leaveWordId);
            request.setAttribute( s: "message",  o: "留言删除成功!");
```



```
            return "message";
        } catch (Exception e) {
            e.printStackTrace();
            request.setAttribute( s: "error",  o: "留言删除失败!");
            return "error";

        }

    }

    /*ajax方式删除多条留言记录*/
    @RequestMapping(value={"/deletes",method=RequestMethod.POST)
    public void delete(String leaveWordIds,HttpServletRequest request,HttpServletResponse response) throws IOException, JSONException {
        String message = "";
        boolean success = false;
        try {
            int count = leavewordService.deleteLeavewords(leaveWordIds);
            success = true;
            message = count + "条记录删除成功";
            writeJsonResponse(response, success, message);
        } catch (Exception e) {
            //e.printStackTrace();
            message = "有记录存在外键约束,删除失败";
            writeJsonResponse(response, success, message);
        }
    }

    /*按照查询条件导出留言信息到Excel*/
    @RequestMapping(value = { "/OutToExcel" }, method = {RequestMethod.GET,RequestMethod.POST})
    public void OutToExcel(String leaveTitle,@ModelAttribute("userObj") UserInfo userObj,String leaveTime, Model model, HttpServletRequest r
            String type = session.getAttribute( s: "type").toString();
            String username = session.getAttribute( s: "username").toString();
            if("2".equals(type)){
                userObj.setUser_name(username);
            }
        if(leaveTitle == null) leaveTitle = "";
        if(leaveTime == null) leaveTime = "";
```



```
        List<Leaveword> leavewordList = leavewordService.queryLeaveword(leaveTitle,userObj,leaveTime);
        ExportExcelUtil ex = new ExportExcelUtil();
        String _title = "Leaveword信息记录";
        String[] headers = { "留言id","留言标题","留言人","留言时间","管理回复","回复时间"};
        List<String[]> dataset = new ArrayList<>();
        for(int i=0;i<leavewordList.size();i++) {
            Leaveword leaveword = leavewordList.get(i);
            dataset.add(new String[]{leaveword.getLeaveWordId() + "",leaveword.getLeaveTitle(),leaveword.getUserObj().getName(),leaveword.g
        }
        /*
        OutputStream out = null;
        try {
            out = new FileOutputStream("C://output.xls");
            ex.exportExcel(title,headers, dataset, out);
            out.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
        */
        OutputStream out = null;//创建一个输出流对象
        try {
            out = response.getOutputStream();//
            response.setHeader( s: "Content-disposition", s1: "attachment; filename="+"Leaveword.xls");//filename是下载的xls的名，建议最好用英文
            response.setContentType("application/msexcel;charset=UTF-8");//设置类型
            response.setHeader( s: "Pragma", s1: "No-cache");//设置头
            response.setHeader( s: "Cache-Control", s1: "no-cache");//设置头
            response.setDateHeader( s: "Expires",  l: 0);//设置日期头
            String rootPath = request.getSession().getServletContext().getRealPath( s: "/");
            ex.exportExcel(rootPath,_title,headers, dataset, out);
            out.flush();
        } catch (IOException e) {
            e.printStackTrace();
        }finally{
            try{
                if(out!=null){
```

```
                    out.close();
                }
            }catch(IOException e){
                e.printStackTrace();
            }
        }
    }
}
```

NewsController：

NewsController.java

```java
package com.chengxusheji.controller;

import java.io.IOException;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.io.UnsupportedEncodingException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.List;

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

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import com.chengxusheji.utils.ExportExcelUtil;
import com.chengxusheji.utils.UserException;
import com.chengxusheji.service.NewsService;
import com.chengxusheji.po.News;

//News管理控制层
@Controller
@RequestMapping("/News")
public class NewsController extends BaseController {
```

```java
    /*业务层对象*/
    17 usages
    @Resource NewsService newsService;

    @InitBinder("news")
    public void initBinderNews(WebDataBinder binder) { binder.setFieldDefaultPrefix("news."); }
    /*跳转到添加News视图*/
    @RequestMapping(value = "/add", method = RequestMethod.GET)
    public String add(Model model,HttpServletRequest request) throws Exception {
        model.addAttribute(new News());
        return "News_add";
    }

    /*客户端ajax方式提交添加新闻信息信息*/
    @RequestMapping(value = "/add", method = RequestMethod.POST)
    public void add(@Validated News news, BindingResult br,
            Model model, HttpServletRequest request,HttpServletResponse response) throws Exception {
        String message = "";
        boolean success = false;
        if (br.hasErrors()) {
            message = "输入信息不符合要求！";
            writeJsonResponse(response, success, message);
            return ;
        }
        try {
            news.setNewsPhoto(this.handlePhotoUpload(request,  fileKeyName: "newsPhotoFile"));
        } catch(UserException ex) {
            message = "图片格式不正确！";
            writeJsonResponse(response, success, message);
            return ;
        }
        newsService.addNews(news);
        message = "新闻信息添加成功!";
        success = true;
        writeJsonResponse(response, success, message);
    }
    /*ajax方式按照查询条件分页查询新闻信息信息*/
    @RequestMapping(value = { "/list" }, method = {RequestMethod.GET,RequestMethod.POST})
    public void list(String newsTitle,String newsDate,Integer page,Integer rows, Model model, HttpServletRequest request,HttpServletResponse
        if (page==null || page == 0) page = 1;
        if (newsTitle == null) newsTitle = "";
        if (newsDate == null) newsDate = "";
        if(rows != 0)newsService.setRows(rows);
        List<News> newsList = newsService.queryNews(newsTitle, newsDate, page);
        /*计算总的页数和总的记录数*/
        newsService.queryTotalPageAndRecordNumber(newsTitle, newsDate);
        /*获取到总的页码数目*/
        int totalPage = newsService.getTotalPage();
        /*当前查询条件下总记录数*/
        int recordNumber = newsService.getRecordNumber();
        response.setContentType("text/json;charset=UTF-8");
        PrintWriter out = response.getWriter();
        //将要被返回到客户端的对象
        JSONObject jsonObj=new JSONObject();
        jsonObj.accumulate("total", recordNumber);
        JSONArray jsonArray = new JSONArray();
        for(News news:newsList) {
            JSONObject jsonNews = news.getJsonObject();
            jsonArray.put(jsonNews);
        }
        jsonObj.accumulate("rows", jsonArray);
        out.println(jsonObj.toString());
        out.flush();
        out.close();
    }

    /*ajax方式按照查询条件分页查询新闻信息信息*/
    @RequestMapping(value = { "/listAll" }, method = {RequestMethod.GET,RequestMethod.POST})
    public void listAll(HttpServletResponse response) throws Exception {
        List<News> newsList = newsService.queryAllNews();
        response.setContentType("text/json;charset=UTF-8");
        PrintWriter out = response.getWriter();
```

```
        JSONArray jsonArray = new JSONArray();
        for(News news:newsList) {
            JSONObject jsonNews = new JSONObject();
            jsonNews.accumulate("newsId", news.getNewsId());
            jsonNews.accumulate("newsTitle", news.getNewsTitle());
            jsonArray.put(jsonNews);
        }
        out.println(jsonArray.toString());
        out.flush();
        out.close();
    }

    /*前台按照查询条件分页查询新闻信息信息*/
    @RequestMapping(value = { "/frontlist" }, method = {RequestMethod.GET,RequestMethod.POST})
    public String frontlist(String newsTitle,String newsDate,Integer currentPage, Model model, HttpServletRequest request) throws Exception
        if (currentPage==null || currentPage == 0) currentPage = 1;
        if (newsTitle == null) newsTitle = "";
        if (newsDate == null) newsDate = "";
        List<News> newsList = newsService.queryNews(newsTitle, newsDate, currentPage);
        /*计算总的页数和总的记录数*/
        newsService.queryTotalPageAndRecordNumber(newsTitle, newsDate);
        /*获取到总的页码数目*/
        int totalPage = newsService.getTotalPage();
        /*当前查询条件下总记录数*/
        int recordNumber = newsService.getRecordNumber();
        request.setAttribute( s "newsList",  newsList);
        request.setAttribute( s "totalPage", totalPage);
        request.setAttribute( s "recordNumber", recordNumber);
        request.setAttribute( s "currentPage", currentPage);
        request.setAttribute( s "newsTitle", newsTitle);
        request.setAttribute( s "newsDate", newsDate);
        return "News/news_frontquery_result";
    }

     /*前台查询News信息*/
    @RequestMapping(value="/{newsId}/frontshow",method=RequestMethod.GET)
```

```
    public String frontshow(@PathVariable Integer newsId,Model model,HttpServletRequest request) throws Exception {
        /*根据主键newsId获取News对象*/
        News news = newsService.getNews(newsId);

        request.setAttribute( s "news",  news);
        return "News/news_frontshow";
    }

    /*ajax方式显示新闻信息修改jsp视图页*/
    @RequestMapping(value="/{newsId}/update",method=RequestMethod.GET)
    public void update(@PathVariable Integer newsId,Model model,HttpServletRequest request,HttpServletResponse response) throws Exception {
        /*根据主键newsId获取News对象*/
        News news = newsService.getNews(newsId);

        response.setContentType("text/json;charset=UTF-8");
        PrintWriter out = response.getWriter();
        //将要被返回到客户端的对象
        JSONObject jsonNews = news.getJsonObject();
        out.println(jsonNews.toString());
        out.flush();
        out.close();
    }

    /*ajax方式更新新闻信息信息*/
    @RequestMapping(value = "/{newsId}/update", method = RequestMethod.POST)
    public void update(@Validated News news, BindingResult br,
            Model model, HttpServletRequest request,HttpServletResponse response) throws Exception {
        String message = "";
        boolean success = false;
        if (br.hasErrors()) {
            message = "输入的信息有错误！";
            writeJsonResponse(response, success, message);
            return;
```

```
        }
        String newsPhotoFileName = this.handlePhotoUpload(request, fileKeyName: "newsPhotoFile");
        if(!newsPhotoFileName.equals("upload/NoImage.jpg"))news.setNewsPhoto(newsPhotoFileName);


        try {
            newsService.updateNews(news);
            message = "新闻信息更新成功!";
            success = true;
            writeJsonResponse(response, success, message);
        } catch (Exception e) {
            e.printStackTrace();
            message = "新闻信息更新失败!";
            writeJsonResponse(response, success, message);
        }
    }
    /*删除新闻信息信息*/
    @RequestMapping(value="/{newsId}/delete",method=RequestMethod.GET)
    public String delete(@PathVariable Integer newsId,HttpServletRequest request) throws UnsupportedEncodingException {
          try {
              newsService.deleteNews(newsId);
                request.setAttribute( s: "message",  o: "新闻信息删除成功!");
                return "message";
            } catch (Exception e) {
                e.printStackTrace();
                request.setAttribute( s: "error",  o: "新闻信息删除失败!");
                return "error";

            }

    }

    /*ajax方式删除多条新闻信息记录*/
    @RequestMapping(value="/deletes",method=RequestMethod.POST)
    public void delete(String newsIds,HttpServletRequest request,HttpServletResponse response) throws IOException, JSONEx
```

```
        String message = "";
        boolean success = false;
        try {
            int count = newsService.deleteNewss(newsIds);
            success = true;
            message = count + "条记录删除成功";
            writeJsonResponse(response, success, message);
        } catch (Exception e) {
            //e.printStackTrace();
            message = "有记录存在外键约束,删除失败";
            writeJsonResponse(response, success, message);
        }
    }

    /*按照查询条件导出新闻信息信息到Excel*/
    @RequestMapping(value = { "/OutToExcel" }, method = {RequestMethod.GET,RequestMethod.POST})
    public void OutToExcel(String newsTitle,String newsDate, Model model, HttpServletRequest request,HttpServletResponse response) throws
        if(newsTitle == null) newsTitle = "";
        if(newsDate == null) newsDate = "";
        List<News> newsList = newsService.queryNews(newsTitle,newsDate);
        ExportExcelUtil ex = new ExportExcelUtil();
        String _title = "News信息记录";
        String[] headers = { "新闻id","新闻标题","新闻图片","新闻日期","新闻来源"};
        List<String[]> dataset = new ArrayList<>();
        for(int i=0;i<newsList.size();i++) {
            News news = newsList.get(i);
            dataset.add(new String[]{news.getNewsId() + "",news.getNewsTitle(),news.getNewsPhoto(),news.getNewsDate(),news.getNewsFrom()});
        }
        /*
        OutputStream out = null;
        try {
            out = new FileOutputStream("C://output.xls");
            ex.exportExcel(title,headers, dataset, out);
            out.close();
        } catch (Exception e) {
            e.printStackTrace();
```

```
        }
        */
        OutputStream out = null;//创建一个输出流对象
        try {
            out = response.getOutputStream();//
            response.setHeader( s: "Content-disposition", s1: "attachment; filename="+"News.xls");//filename是下载的xls的名，建议最好用英文
            response.setContentType("application/msexcel;charset=UTF-8");//设置类型
            response.setHeader( s: "Pragma", s1: "No-cache");//设置头
            response.setHeader( s: "Cache-Control", s1: "no-cache");//设置头
            response.setDateHeader( s: "Expires",  l: 0);//设置日期头
            String rootPath = request.getSession().getServletContext().getRealPath( s: "/");
            ex.exportExcel(rootPath,_title,headers, dataset, out);
            out.flush();
        } catch (IOException e) {
            e.printStackTrace();
        }finally{
            try{
                if(out!=null){
                    out.close();
                }
            }catch(IOException e){
                e.printStackTrace();
            }
        }
    }
}
```

OrderInfoController:

OrderInfoController.java

```
package com.chengxusheji.controller;

import java.io.IOException;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.io.UnsupportedEncodingException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.List;

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

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import com.chengxusheji.utils.ExportExcelUtil;
import com.chengxusheji.utils.UserException;
import com.chengxusheji.service.OrderInfoService;
import com.chengxusheji.po.OrderInfo;
import com.chengxusheji.service.DoctorService;
import com.chengxusheji.po.Doctor;
import com.chengxusheji.service.UserInfoService;
import com.chengxusheji.po.UserInfo;

```

```java
//OrderInfo管理控制层
@Controller
@RequestMapping("/OrderInfo")
public class OrderInfoController extends BaseController {

    /*业务层对象*/
    17 usages
    @Resource OrderInfoService orderInfoService;

    3 usages
    @Resource DoctorService doctorService;
    3 usages
    @Resource UserInfoService userInfoService;
    @InitBinder("userObj")
    public void initBinderuserObj(WebDataBinder binder) { binder.setFieldDefaultPrefix("userObj."); }
    @InitBinder("doctorObj")
    public void initBinderdoctorObj(WebDataBinder binder) { binder.setFieldDefaultPrefix("doctorObj."); }
    @InitBinder("orderInfo")
    public void initBinderOrderInfo(WebDataBinder binder) { binder.setFieldDefaultPrefix("orderInfo."); }
    /*跳转到添加OrderInfo视图*/
    @RequestMapping(value = "/add", method = RequestMethod.GET)
    public String add(Model model,HttpServletRequest request) throws Exception {
        model.addAttribute(new OrderInfo());
        /*查询所有的Doctor信息*/
        List<Doctor> doctorList = doctorService.queryAllDoctor();
        request.setAttribute( s: "doctorList", doctorList);
        /*查询所有的UserInfo信息*/
        List<UserInfo> userInfoList = userInfoService.queryAllUserInfo();
        request.setAttribute( s: "userInfoList", userInfoList);
        return "OrderInfo_add";
    }

    /*客户端ajax方式提交添加预约信息信息*/
    @RequestMapping(value = "/add", method = RequestMethod.POST)
    public void add(@Validated OrderInfo orderInfo, BindingResult br,
            Model model, HttpServletRequest request,HttpServletResponse response) throws Exception {
        String message = "";
        boolean success = false;
        if (br.hasErrors()) {
            message = "输入信息不符合要求！";
            writeJsonResponse(response, success, message);
            return ;
        }
        orderInfoService.addOrderInfo(orderInfo);
        message = "预约信息添加成功!";
        success = true;
        writeJsonResponse(response, success, message);
    }
    /*ajax方式按照查询条件分页查询预约信息信息*/
    @RequestMapping(value = { "/list" }, method = {RequestMethod.GET,RequestMethod.POST})
    public void list(@ModelAttribute("userObj") UserInfo userObj,@ModelAttribute("doctorObj") Doctor doctorObj,String orderDate,String timeI
        String type = session.getAttribute( s: "type").toString();
        String username = session.getAttribute( s: "username").toString();
        if("2".equals(type)){
            userObj.setUser_name(username);
        }
        if (page==null || page == 0) page = 1;
        if (orderDate == null) orderDate = "";
        if (timeInterval == null) timeInterval = "";
        if (telephone == null) telephone = "";
        if (checkState == null) checkState = "";
        if (orderTime == null) orderTime = "";
        if(rows != 0)orderInfoService.setRows(rows);
        List<OrderInfo> orderInfoList = orderInfoService.queryOrderInfo(userObj, doctorObj, orderDate, timeInterval, telephone, checkState,
        /*计算总的页数和总的记录数*/
        orderInfoService.queryTotalPageAndRecordNumber(userObj, doctorObj, orderDate, timeInterval, telephone, checkState, orderTime);
        /*获取到总的页码数目*/
        int totalPage = orderInfoService.getTotalPage();
        /*当前查询条件下总记录数*/
        int recordNumber = orderInfoService.getRecordNumber();
        response.setContentType("text/json;charset=UTF-8");
        PrintWriter out = response.getWriter();
        //将要被返回到客户端的对象
```

```
        JSONObject jsonObj=new JSONObject();
        jsonObj.accumulate("total", recordNumber);
        JSONArray jsonArray = new JSONArray();
        for(OrderInfo orderInfo:orderInfoList) {
            JSONObject jsonOrderInfo = orderInfo.getJsonObject();
            jsonArray.put(jsonOrderInfo);
        }
        jsonObj.accumulate("rows", jsonArray);
        out.println(jsonObj.toString());
        out.flush();
        out.close();
    }

    /*ajax方式按照查询条件分页查询预约信息信息*/
    @RequestMapping(value = { "/listAll" }, method = {RequestMethod.GET,RequestMethod.POST})
    public void listAll(HttpServletResponse response) throws Exception {
        List<OrderInfo> orderInfoList = orderInfoService.queryAllOrderInfo();
        response.setContentType("text/json;charset=UTF-8");
        PrintWriter out = response.getWriter();
        JSONArray jsonArray = new JSONArray();
        for(OrderInfo orderInfo:orderInfoList) {
            JSONObject jsonOrderInfo = new JSONObject();
            jsonOrderInfo.accumulate("orderId", orderInfo.getOrderId());
            jsonArray.put(jsonOrderInfo);
        }
        out.println(jsonArray.toString());
        out.flush();
        out.close();
    }

    /*前台按照查询条件分页查询预约信息信息*/
    @RequestMapping(value = { "/frontlist" }, method = {RequestMethod.GET,RequestMethod.POST})
    public String frontlist(@ModelAttribute("userObj") UserInfo userObj,@ModelAttribute("doctorObj") Doctor doctorObj,String orderDate,Strin
        String type = session.getAttribute( s: "type").toString();
        String username = session.getAttribute( s: "username").toString();
        if("2".equals(type)){
            userObj.setUser_name(username);
        }
        if (currentPage==null || currentPage == 0) currentPage = 1;
        if (orderDate == null) orderDate = "";
        if (timeInterval == null) timeInterval = "";
        if (telephone == null) telephone = "";
        if (checkState == null) checkState = "";
        if (orderTime == null) orderTime = "";
        List<OrderInfo> orderInfoList = orderInfoService.queryOrderInfo(userObj, doctorObj, orderDate, timeInterval, telephone, checkState,
        /*计算总的页数和总的记录数*/
        orderInfoService.queryTotalPageAndRecordNumber(userObj, doctorObj, orderDate, timeInterval, telephone, checkState, orderTime);
        /*获取到总的页码数目*/
        int totalPage = orderInfoService.getTotalPage();
        /*当前查询条件下总记录数*/
        int recordNumber = orderInfoService.getRecordNumber();
        request.setAttribute( s: "orderInfoList",  orderInfoList);
        request.setAttribute( s: "totalPage", totalPage);
        request.setAttribute( s: "recordNumber", recordNumber);
        request.setAttribute( s: "currentPage", currentPage);
        request.setAttribute( s: "userObj", userObj);
        request.setAttribute( s: "doctorObj", doctorObj);
        request.setAttribute( s: "orderDate", orderDate);
        request.setAttribute( s: "timeInterval", timeInterval);
        request.setAttribute( s: "telephone", telephone);
        request.setAttribute( s: "checkState", checkState);
        request.setAttribute( s: "orderTime", orderTime);
        List<Doctor> doctorList = doctorService.queryAllDoctor();
        request.setAttribute( s: "doctorList", doctorList);
        List<UserInfo> userInfoList = userInfoService.queryAllUserInfo();
        request.setAttribute( s: "userInfoList", userInfoList);
        return "OrderInfo/orderInfo_frontquery_result";
    }

     /*前台查询OrderInfo信息*/
    @RequestMapping(value="/{orderId}/frontshow",method=RequestMethod.GET)
    public String frontshow(@PathVariable Integer orderId,Model model,HttpServletRequest request) throws Exception {
```

```
        /*根据主键orderId获取OrderInfo对象*/
        OrderInfo orderInfo = orderInfoService.getOrderInfo(orderId);

        List<Doctor> doctorList = doctorService.queryAllDoctor();
        request.setAttribute( s: "doctorList", doctorList);
        List<UserInfo> userInfoList = userInfoService.queryAllUserInfo();
        request.setAttribute( s: "userInfoList", userInfoList);
        request.setAttribute( s: "orderInfo",  orderInfo);
        return "OrderInfo/orderInfo_frontshow";
    }

    /*ajax方式显示预约信息修改jsp视图页*/
    @RequestMapping(value="/{orderId}/update",method=RequestMethod.GET)
    public void update(@PathVariable Integer orderId,Model model,HttpServletRequest request,HttpServletResponse response) throws Exception {
        /*根据主键orderId获取OrderInfo对象*/
        OrderInfo orderInfo = orderInfoService.getOrderInfo(orderId);

        response.setContentType("text/json;charset=UTF-8");
        PrintWriter out = response.getWriter();
        //将要被返回到客户端的对象
        JSONObject jsonOrderInfo = orderInfo.getJsonObject();
        out.println(jsonOrderInfo.toString());
        out.flush();
        out.close();
    }

    /*ajax方式更新预约信息信息*/
    @RequestMapping(value = "/{orderId}/update", method = RequestMethod.POST)
    public void update(@Validated OrderInfo orderInfo, BindingResult br,
            Model model, HttpServletRequest request,HttpServletResponse response) throws Exception {
        String message = "";
        boolean success = false;
        if (br.hasErrors()) {
            message = "输入的信息有错误！";
            writeJsonResponse(response, success, message);
```

```
            return;
        }
        try {
            orderInfoService.updateOrderInfo(orderInfo);
            message = "预约信息更新成功!";
            success = true;
            writeJsonResponse(response, success, message);
        } catch (Exception e) {
            e.printStackTrace();
            message = "预约信息更新失败!";
            writeJsonResponse(response, success, message);
        }
    }
    /*删除预约信息信息*/
    @RequestMapping(value="/{orderId}/delete",method=RequestMethod.GET)
    public String delete(@PathVariable Integer orderId,HttpServletRequest request) throws UnsupportedEncodingException {
          try {
              orderInfoService.deleteOrderInfo(orderId);
                request.setAttribute( s: "message",  o: "预约信息删除成功!");
                return "message";
            } catch (Exception e) {
                e.printStackTrace();
                request.setAttribute( s: "error",  o: "预约信息删除失败!");
                return "error";

            }

    }

    /*ajax方式删除多条预约信息记录*/
    @RequestMapping(value="/deletes",method=RequestMethod.POST)
    public void delete(String orderIds,HttpServletRequest request,HttpServletResponse response) throws IOException, JSONException {
        String message = "";
        boolean success = false;
        try {
            int count = orderInfoService.deleteOrderInfos(orderIds);
            success = true;
```

```
            message = count + "条记录删除成功";
            writeJsonResponse(response, success, message);
        } catch (Exception e) {
            //e.printStackTrace();
            message = "有记录存在外键约束,删除失败";
            writeJsonResponse(response, success, message);
        }
    }

    /*按照查询条件导出预约信息信息到Excel*/
    @RequestMapping(value = { "/OutToExcel" }, method = {RequestMethod.GET,RequestMethod.POST})
    public void OutToExcel(@ModelAttribute("userObj") UserInfo userObj,@ModelAttribute("doctorObj") Doctor doctorObj,String orderDate,String
        if(orderDate == null) orderDate = "";
        if(timeInterval == null) timeInterval = "";
        if(telephone == null) telephone = "";
        if(checkState == null) checkState = "";
        if(orderTime == null) orderTime = "";
                String type = session.getAttribute( s: "type").toString();
                String username = session.getAttribute( s: "username").toString();
                if("2".equals(type)){
                    userObj.setUser_name(username);
                }
        List<OrderInfo> orderInfoList = orderInfoService.queryOrderInfo(userObj,doctorObj,orderDate,timeInterval,telephone,checkState,orderT
        ExportExcelUtil ex = new ExportExcelUtil();
        String _title = "OrderInfo信息记录";
        String[] headers = { "预约id","预约用户","预约医生","预约日期","时段","联系电话","下单时间","处理状态","医生回复"};
        List<String[]> dataset = new ArrayList<>();
        for(int i=0;i<orderInfoList.size();i++) {
            OrderInfo orderInfo = orderInfoList.get(i);
            dataset.add(new String[]{orderInfo.getOrderId() + "",orderInfo.getUserObj().getName(),orderInfo.getDoctorObj().getDoctorName(),o
        }
        /*
        OutputStream out = null;
        try {
            out = new FileOutputStream("C://output.xls");
            ex.exportExcel(title,headers, dataset, out);
            out.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
        */
        OutputStream out = null;//创建一个输出流对象
        try {
            out = response.getOutputStream();//
            response.setHeader( s: "Content-disposition", s1: "attachment; filename="+"OrderInfo.xls");//filename是下载的xls的名，建议最好用英文
            response.setContentType("application/msexcel;charset=UTF-8");//设置类型
            response.setHeader( s: "Pragma", s1: "No-cache");//设置头
            response.setHeader( s: "Cache-Control", s1: "no-cache");//设置头
            response.setDateHeader( s: "Expires",  l: 0);//设置日期头
            String rootPath = request.getSession().getServletContext().getRealPath( s: "/");
            ex.exportExcel(rootPath,_title,headers, dataset, out);
            out.flush();
        } catch (IOException e) {
            e.printStackTrace();
        }finally{
            try{
                if(out!=null){
                    out.close();
                }
            }catch(IOException e){
                e.printStackTrace();
            }
        }
    }
}
```

PatientController:

PatientController.java

```java
package com.chengxusheji.controller;

import java.io.IOException;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.io.UnsupportedEncodingException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.List;

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

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import com.chengxusheji.utils.ExportExcelUtil;
import com.chengxusheji.utils.UserException;
import com.chengxusheji.service.PatientService;
import com.chengxusheji.po.Patient;
import com.chengxusheji.service.DoctorService;
import com.chengxusheji.po.Doctor;

//Patient管理控制层
@Controller
@RequestMapping("/Patient")
public class PatientController extends BaseController {

    /*业务层对象*/
    17 usages
    @Resource PatientService patientService;

    3 usages
    @Resource DoctorService doctorService;
    @InitBinder("doctorObj")
    public void initBinderdoctorObj(WebDataBinder binder) { binder.setFieldDefaultPrefix("doctorObj."); }
    @InitBinder("patient")
    public void initBinderPatient(WebDataBinder binder) { binder.setFieldDefaultPrefix("patient."); }
    /*跳转到添加Patient视图*/
    @RequestMapping(value = "/add", method = RequestMethod.GET)
    public String add(Model model,HttpServletRequest request) throws Exception {
        model.addAttribute(new Patient());
        /*查询所有的Doctor信息*/
        List<Doctor> doctorList = doctorService.queryAllDoctor();
        request.setAttribute( s: "doctorList", doctorList);
        return "Patient_add";
    }

    /*客户端ajax方式提交添加病人信息信息*/
    @RequestMapping(value = "/add", method = RequestMethod.POST)
    public void add(@Validated Patient patient, BindingResult br,
            Model model, HttpServletRequest request,HttpServletResponse response) throws Exception {
        String message = "";
        boolean success = false;
        if (br.hasErrors()) {
            message = "输入信息不符合要求！";
            writeJsonResponse(response, success, message);
            return ;
        }
        patientService.addPatient(patient);
        message = "病人信息添加成功!";
        success = true;
```

```
        writeJsonResponse(response, success, message);
    }
    /*ajax方式按照查询条件分页查询病人信息信息*/
    @RequestMapping(value = { "/list" }, method = {RequestMethod.GET,RequestMethod.POST})
    public void list(@ModelAttribute("doctorObj") Doctor doctorObj,String patientName,String cardNumber,String telephone,String addTime,Inte
        if (page==null || page == 0) page = 1;
        if (patientName == null) patientName = "";
        if (cardNumber == null) cardNumber = "";
        if (telephone == null) telephone = "";
        if (addTime == null) addTime = "";
        if(rows != 0)patientService.setRows(rows);
        List<Patient> patientList = patientService.queryPatient(doctorObj, patientName, cardNumber, telephone, addTime, page);
        /*计算总的页数和总的记录数*/
        patientService.queryTotalPageAndRecordNumber(doctorObj, patientName, cardNumber, telephone, addTime);
        /*获取到总的页码数目*/
        int totalPage = patientService.getTotalPage();
        /*当前查询条件下总记录数*/
        int recordNumber = patientService.getRecordNumber();
        response.setContentType("text/json;charset=UTF-8");
        PrintWriter out = response.getWriter();
        //将要被返回到客户端的对象
        JSONObject jsonObj=new JSONObject();
        jsonObj.accumulate("total", recordNumber);
        JSONArray jsonArray = new JSONArray();
        for(Patient patient:patientList) {
            JSONObject jsonPatient = patient.getJsonObject();
            jsonArray.put(jsonPatient);
        }
        jsonObj.accumulate("rows", jsonArray);
        out.println(jsonObj.toString());
        out.flush();
        out.close();
    }

    /*ajax方式按照查询条件分页查询病人信息信息*/
    @RequestMapping(value = { "/listAll" }, method = {RequestMethod.GET,RequestMethod.POST})
```

```
    public void listAll(HttpServletResponse response) throws Exception {
        List<Patient> patientList = patientService.queryAllPatient();
        response.setContentType("text/json;charset=UTF-8");
        PrintWriter out = response.getWriter();
        JSONArray jsonArray = new JSONArray();
        for(Patient patient:patientList) {
            JSONObject jsonPatient = new JSONObject();
            jsonPatient.accumulate("patientId", patient.getPatientId());
            jsonPatient.accumulate("patientName", patient.getPatientName());
            jsonArray.put(jsonPatient);
        }
        out.println(jsonArray.toString());
        out.flush();
        out.close();
    }

    /*前台按照查询条件分页查询病人信息信息*/
    @RequestMapping(value = { "/frontlist" }, method = {RequestMethod.GET,RequestMethod.POST})
    public String frontlist(@ModelAttribute("doctorObj") Doctor doctorObj,String patientName,String cardNumber,String telephone,String addTi
        if (currentPage==null || currentPage == 0) currentPage = 1;
        if (patientName == null) patientName = "";
        if (cardNumber == null) cardNumber = "";
        if (telephone == null) telephone = "";
        if (addTime == null) addTime = "";
        List<Patient> patientList = patientService.queryPatient(doctorObj, patientName, cardNumber, telephone, addTime, currentPage);
        /*计算总的页数和总的记录数*/
        patientService.queryTotalPageAndRecordNumber(doctorObj, patientName, cardNumber, telephone, addTime);
        /*获取到总的页码数目*/
        int totalPage = patientService.getTotalPage();
        /*当前查询条件下总记录数*/
        int recordNumber = patientService.getRecordNumber();
        request.setAttribute( s: "patientList",  patientList);
        request.setAttribute( s: "totalPage", totalPage);
        request.setAttribute( s: "recordNumber", recordNumber);
        request.setAttribute( s: "currentPage", currentPage);
        request.setAttribute( s: "doctorObj", doctorObj);
```

```
        request.setAttribute( s: "patientName", patientName);
        request.setAttribute( s: "cardNumber", cardNumber);
        request.setAttribute( s: "telephone", telephone);
        request.setAttribute( s: "addTime", addTime);
        List<Doctor> doctorList = doctorService.queryAllDoctor();
        request.setAttribute( s: "doctorList", doctorList);
        return "Patient/patient_frontquery_result";
    }

     /*前台查询Patient信息*/
    @RequestMapping(value="/{patientId}/frontshow",method=RequestMethod.GET)
    public String frontshow(@PathVariable Integer patientId,Model model,HttpServletRequest request) throws Exception {
        /*根据主键patientId获取Patient对象*/
        Patient patient = patientService.getPatient(patientId);

        List<Doctor> doctorList = doctorService.queryAllDoctor();
        request.setAttribute( s: "doctorList", doctorList);
        request.setAttribute( s: "patient",  patient);
        return "Patient/patient_frontshow";
    }

    /*ajax方式显示病人信息修改jsp视图页*/
    @RequestMapping(value="/{patientId}/update",method=RequestMethod.GET)
    public void update(@PathVariable Integer patientId,Model model,HttpServletRequest request,HttpServletResponse response) throws Exception
        /*根据主键patientId获取Patient对象*/
        Patient patient = patientService.getPatient(patientId);

        response.setContentType("text/json;charset=UTF-8");
        PrintWriter out = response.getWriter();
        //将要被返回到客户端的对象
        JSONObject jsonPatient = patient.getJsonObject();
        out.println(jsonPatient.toString());
        out.flush();
```

```
        out.close();
    }

     /*ajax方式更新病人信息信息*/
    @RequestMapping(value = "/{patientId}/update", method = RequestMethod.POST)
    public void update(@Validated Patient patient, BindingResult br,
            Model model, HttpServletRequest request,HttpServletResponse response) throws Exception {
        String message = "";
        boolean success = false;
        if (br.hasErrors()) {
            message = "输入的信息有错误！";
            writeJsonResponse(response, success, message);
            return;
        }
        try {
            patientService.updatePatient(patient);
            message = "病人信息更新成功!";
            success = true;
            writeJsonResponse(response, success, message);
        } catch (Exception e) {
            e.printStackTrace();
            message = "病人信息更新失败!";
            writeJsonResponse(response, success, message);
        }
    }
    /*删除病人信息信息*/
    @RequestMapping(value="/{patientId}/delete",method=RequestMethod.GET)
    public String delete(@PathVariable Integer patientId,HttpServletRequest request) throws UnsupportedEncodingException {
          try {
              patientService.deletePatient(patientId);
                request.setAttribute( s: "message",  o: "病人信息删除成功!");
                return "message";
            } catch (Exception e) {
                e.printStackTrace();
                request.setAttribute( s: "error",  o: "病人信息删除失败!");
```

```
                return "error";

            }

    }

    /*ajax方式删除多条病人信息记录*/
    @RequestMapping(value="/deletes",method=RequestMethod.POST)
    public void delete(String patientIds,HttpServletRequest request,HttpServletResponse response) throws IOException, JSONException {
        String message = "";
        boolean success = false;
        try {
            int count = patientService.deletePatients(patientIds);
            success = true;
            message = count + "条记录删除成功";
            writeJsonResponse(response, success, message);
        } catch (Exception e) {
            //e.printStackTrace();
            message = "有记录存在外键约束,删除失败";
            writeJsonResponse(response, success, message);
        }
    }

    /*按照查询条件导出病人信息信息到Excel*/
    @RequestMapping(value = { "/OutToExcel" }, method = {RequestMethod.GET,RequestMethod.POST})
    public void OutToExcel(@ModelAttribute("doctorObj") Doctor doctorObj,String patientName,String cardNumber,String telephone,String addTi
        if(patientName == null) patientName = "";
        if(cardNumber == null) cardNumber = "";
        if(telephone == null) telephone = "";
        if(addTime == null) addTime = "";
        List<Patient> patientList = patientService.queryPatient(doctorObj,patientName,cardNumber,telephone,addTime);
        ExportExcelUtil ex = new ExportExcelUtil();
        String _title = "Patient信息记录";
```

```
        String[] headers = { "病人id","医生","病人姓名","性别","身份证号","联系电话","登记时间"};
        List<String[]> dataset = new ArrayList<>();
        for(int i=0;i<patientList.size();i++) {
            Patient patient = patientList.get(i);
            dataset.add(new String[]{patient.getPatientId() + "",patient.getDoctorObj().getDoctorName(),patient.getPatientName(),patient.
        }
        /*
        OutputStream out = null;
        try {
            out = new FileOutputStream("C://output.xls");
            ex.exportExcel(title,headers, dataset, out);
            out.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
        */
```

```
        OutputStream out = null;//创建一个输出流对象
        try {
            out = response.getOutputStream();//
            response.setHeader( s: "Content-disposition", s1: "attachment; filename="+"Patient.xls");//filename是下载的xls的名，建议最好用英文
            response.setContentType("application/msexcel;charset=UTF-8");//设置类型
            response.setHeader( s: "Pragma", s1: "No-cache");//设置头
            response.setHeader( s: "Cache-Control", s1: "no-cache");//设置头
            response.setDateHeader( s: "Expires", l: 0);//设置日期头
            String rootPath = request.getSession().getServletContext().getRealPath( s: "/");
            ex.exportExcel(rootPath,_title,headers, dataset, out);
            out.flush();
        } catch (IOException e) {
            e.printStackTrace();
        }finally{
            try{
                if(out!=null){
                    out.close();
                }
            }catch(IOException e){
                e.printStackTrace();
            }
        }
    }
}
```

SystemController:

SystemController.java

```java
package com.chengxusheji.controller;

import com.chengxusheji.po.UserInfo;
import com.chengxusheji.service.UserInfoService;
import java.io.PrintWriter;

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

import org.json.JSONObject;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.validation.annotation.Validated;
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.SessionAttributes;


import com.chengxusheji.po.Admin;
import com.chengxusheji.service.AdminService;
import com.chengxusheji.utils.UserException;


@Controller
@SessionAttributes("username")
public class SystemController  extends BaseController{

    7 usages
    @Resource AdminService adminService;
    no usages
    @Resource
    UserInfoService userInfoService;

    @RequestMapping(value="/login",method=RequestMethod.GET)
    public String login(Model model) {
        model.addAttribute(new Admin());
        return "login";
    }

    //前台用户登录
    @RequestMapping(value="/frontLogin",method=RequestMethod.POST)
    public void frontLogin(@RequestParam("userName")String userName,@RequestParam("password")String password,HttpServletResponse response,Ht
        boolean success = true;
        String msg = "";
        Admin admin = new Admin();
        admin.setUsername(userName);
        admin.setPassword(password);
        admin   = adminService.checkLogin(admin);
        if (admin == null) {
            msg = "用户不存在或密码错误!";
            success = false;
        }
        if(success) {
            session.setAttribute( 实 "user_name", userName);
            session.setAttribute( 实 "username", userName);
        }

        response.setContentType("text/json;charset=GBK");
        PrintWriter out = response.getWriter();
        //将要被返回到客户端的对象
        JSONObject json=new JSONObject();
        json.accumulate("success", success);
        json.accumulate("msg", msg);
        out.println(json.toString());
        out.flush();
        out.close();
    }


    @RequestMapping(value="/login",method=RequestMethod.POST)
```

```java
public void login(@Validated Admin admin,BindingResult br,Model model,HttpServletRequest request,HttpServletResponse response,HttpSessio
    boolean success = true;
    String msg = "";
    if(br.hasErrors()) {
        msg = br.getAllErrors().toString();
        success = false;
    }
    Admin user = adminService.checkLogin(admin);
    if (user==null) {
        msg = adminService.getErrMessage();
        success = false;
    }
    if(success) {
        session.setAttribute( s: "username", user.getUsername());
        session.setAttribute( s: "type", user.getType());
    }
    response.setContentType("text/json;charset=UTF-8");
    PrintWriter out = response.getWriter();
    //将要被返回到客户端的对象
    JSONObject json=new JSONObject();
    json.accumulate("success", success);
    json.accumulate("msg", msg);
    out.println(json.toString());
    out.flush();
    out.close();
}



@RequestMapping("/logout")
public String logout(Model model,HttpSession session) {
    model.asMap().remove( key: "username");
    session.invalidate();
    return "redirect:/";
}
```

```java


@RequestMapping(value="/changePassword",method=RequestMethod.POST)
public String ChangePassword(String oldPassword,String newPassword,String newPassword2,HttpServletRequest request,HttpSession session) t
    if(oldPassword.equals("")) throw new UserException("请输入旧密码！");
    if(newPassword.equals("")) throw new UserException("请输入新密码！");
    if(!newPassword.equals(newPassword2)) throw new UserException("两次新密码输入不一致");

    String username = (String)session.getAttribute( s: "username");
    if(username == null) throw new UserException("session会话超时，请重新登录系统!");


    Admin admin = adminService.findAdminByUserName(username);
    if(!admin.getPassword().equals(oldPassword)) throw new UserException("输入的旧密码不正确!");

    try {
        adminService.changePassword(username,newPassword);
        request.setAttribute( s: "message", java.net.URLEncoder.encode(
                s: "密码修改成功!", enc: "GBK"));
        return "message";
    } catch (Exception e) {
        e.printStackTrace();
        request.setAttribute( s: "error", java.net.URLEncoder
                .encode( s: "密码修改失败!", enc: "GBK"));
        return "error";
    }
}

@RequestMapping(value = "/register", method = RequestMethod.POST)
public void register(@Validated Admin admin, BindingResult br,
        Model model, HttpServletRequest request,HttpServletResponse response) throws Exception {
    String message = "";
    boolean success = false;
    if (br.hasErrors()) {
        message = "输入信息不符合要求！";
        writeJsonResponse(response, success, message);
```

```
            return ;
        }
        if(adminService.findAdminByUserName(admin.getUsername()) != null) {
            message = "用户名已经存在！";
            writeJsonResponse(response, success, message);
            return ;
        }
        adminService.addAdmin(admin);
        message = "用户添加成功!";
        success = true;
        writeJsonResponse(response, success, message);
    }


}

```

UserInfoController:

UserInfoController.java

```
package com.chengxusheji.controller;

import com.chengxusheji.po.Admin;
import com.chengxusheji.service.AdminService;
import java.io.IOException;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.io.UnsupportedEncodingException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.List;

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

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import com.chengxusheji.utils.ExportExcelUtil;
import com.chengxusheji.utils.UserException;
import com.chengxusheji.service.UserInfoService;
import com.chengxusheji.po.UserInfo;

//UserInfo管理控制层
@Controller
@RequestMapping("/UserInfo")
```

```java
@RequestMapping("/UserInfo")
public class UserInfoController extends BaseController {

    3 usages
    @Resource
    AdminService adminService;
    /*业务层对象*/
    18 usages
    @Resource UserInfoService userInfoService;

    @InitBinder("userInfo")
    public void initBinderUserInfo(WebDataBinder binder) { binder.setFieldDefaultPrefix("userInfo."); }
    /*跳转到添加UserInfo视图*/
    @RequestMapping(value = "/add", method = RequestMethod.GET)
    public String add(Model model,HttpServletRequest request) throws Exception {
        model.addAttribute(new UserInfo());
        return "UserInfo_add";
    }

    /*客户端ajax方式提交添加用户信息*/
    @RequestMapping(value = "/add", method = RequestMethod.POST)
    public void add(@Validated UserInfo userInfo, BindingResult br,
            Model model, HttpServletRequest request,HttpServletResponse response) throws Exception {
        String message = "";
        boolean success = false;
        if (br.hasErrors()) {
            message = "输入信息不符合要求！";
            writeJsonResponse(response, success, message);
            return ;
        }
        if(userInfoService.getUserInfo(userInfo.getUser_name()) != null) {
            message = "用户名已经存在！";
            writeJsonResponse(response, success, message);
            return ;
        }
        Admin admin =adminService.findAdminByUserName(userInfo.getUser_name());
        if(admin != null) {
            message = "用户名已经存在！";
            writeJsonResponse(response, success, message);
            return ;
        }
        try {
            userInfo.setUserPhoto(this.handlePhotoUpload(request, fileKeyName: "userPhotoFile"));
        } catch(UserException ex) {
            message = "图片格式不正确！";
            writeJsonResponse(response, success, message);
            return ;
        }
            admin = new Admin();
            admin.setPassword(userInfo.getPassword());
            admin.setUsername(userInfo.getUser_name());
            admin.setType(2);
            adminService.addAdmin(admin);
            admin = adminService.findAdminByUserName(userInfo.getUser_name());
            userInfo.setAdminid(admin.getId());
            userInfoService.addUserInfo(userInfo);
        message = "用户添加成功!";
        success = true;
        writeJsonResponse(response, success, message);
    }
    /*ajax方式按照查询条件分页查询用户信息*/
    @RequestMapping(value = { "/list" }, method = {RequestMethod.GET,RequestMethod.POST})
    public void list(String user_name,String name,String birthDate,String telephone,Integer page,Integer rows, Model model, HttpServletReque
        if (page==null || page == 0) page = 1;
        if (user_name == null) user_name = "";
        if (name == null) name = "";
        if (birthDate == null) birthDate = "";
        if (telephone == null) telephone = "";
        if(rows != 0)userInfoService.setRows(rows);
        List<UserInfo> userInfoList = userInfoService.queryUserInfo(user_name, name, birthDate, telephone, page);
        /*计算总的页数和总的记录数*/
        userInfoService.queryTotalPageAndRecordNumber(user_name, name, birthDate, telephone);
```

```
        /*获取到总的页码数目*/
        int totalPage = userInfoService.getTotalPage();
        /*当前查询条件下总记录数*/
        int recordNumber = userInfoService.getRecordNumber();
        response.setContentType("text/json;charset=UTF-8");
        PrintWriter out = response.getWriter();
        //将要被返回到客户端的对象
        JSONObject jsonObj=new JSONObject();
        jsonObj.accumulate("total", recordNumber);
        JSONArray jsonArray = new JSONArray();
        for(UserInfo userInfo:userInfoList) {
            JSONObject jsonUserInfo = userInfo.getJsonObject();
            jsonArray.put(jsonUserInfo);
        }
        jsonObj.accumulate("rows", jsonArray);
        out.println(jsonObj.toString());
        out.flush();
        out.close();
    }

    /*ajax方式按照查询条件分页查询用户信息*/
    @RequestMapping(value = { "/listAll" }, method = {RequestMethod.GET,RequestMethod.POST})
    public void listAll(HttpServletResponse response) throws Exception {
        List<UserInfo> userInfoList = userInfoService.queryAllUserInfo();
        response.setContentType("text/json;charset=UTF-8");
        PrintWriter out = response.getWriter();
        JSONArray jsonArray = new JSONArray();
        for(UserInfo userInfo:userInfoList) {
            JSONObject jsonUserInfo = new JSONObject();
            jsonUserInfo.accumulate("user_name", userInfo.getUser_name());
            jsonUserInfo.accumulate("name", userInfo.getName());
            jsonArray.put(jsonUserInfo);
        }
        out.println(jsonArray.toString());
```

```
        out.flush();
        out.close();
    }

    /*前台按照查询条件分页查询用户信息*/
    @RequestMapping(value = { "/frontlist" }, method = {RequestMethod.GET,RequestMethod.POST})
    public String frontlist(String user_name,String name,String birthDate,String telephone,Integer currentPage, Model model, HttpServletRequ
        if (currentPage==null || currentPage == 0) currentPage = 1;
        if (user_name == null) user_name = "";
        if (name == null) name = "";
        if (birthDate == null) birthDate = "";
        if (telephone == null) telephone = "";
        List<UserInfo> userInfoList = userInfoService.queryUserInfo(user_name, name, birthDate, telephone, currentPage);
        /*计算总的页数和总的记录数*/
        userInfoService.queryTotalPageAndRecordNumber(user_name, name, birthDate, telephone);
        /*获取到总的页码数目*/
        int totalPage = userInfoService.getTotalPage();
        /*当前查询条件下总记录数*/
        int recordNumber = userInfoService.getRecordNumber();
        request.setAttribute( s: "userInfoList",  userInfoList);
        request.setAttribute( s: "totalPage", totalPage);
        request.setAttribute( s: "recordNumber", recordNumber);
        request.setAttribute( s: "currentPage", currentPage);
        request.setAttribute( s: "user_name", user_name);
        request.setAttribute( s: "name", name);
        request.setAttribute( s: "birthDate", birthDate);
        request.setAttribute( s: "telephone", telephone);
        return "UserInfo/userInfo_frontquery_result";
    }

     /*前台查询UserInfo信息*/
    @RequestMapping(value="/{user_name}/frontshow",method=RequestMethod.GET)
    public String frontshow(@PathVariable String user_name,Model model,HttpServletRequest request) throws Exception {
        /*根据主键user_name获取UserInfo对象*/
        UserInfo userInfo = userInfoService.getUserInfo(user_name);
```

```
        request.setAttribute( s: "userInfo",  userInfo);
        return "UserInfo/userInfo_frontshow";
    }

    /*ajax方式显示用户修改jsp视图页*/
    @RequestMapping(value="/{user_name}/update",method=RequestMethod.GET)
    public void update(@PathVariable String user_name,Model model,HttpServletRequest request,HttpServletResponse response) throws Exception
        /*根据主键user_name获取UserInfo对象*/
        UserInfo userInfo = userInfoService.getUserInfo(user_name);

        response.setContentType("text/json;charset=UTF-8");
        PrintWriter out = response.getWriter();
        //将要被返回到客户端的对象
        JSONObject jsonUserInfo = userInfo.getJsonObject();
        out.println(jsonUserInfo.toString());
        out.flush();
        out.close();
    }

    /*ajax方式更新用户信息*/
    @RequestMapping(value = "/{user_name}/update", method = RequestMethod.POST)
    public void update(@Validated UserInfo userInfo, BindingResult br,
            Model model, HttpServletRequest request,HttpServletResponse response) throws Exception {
        String message = "";
        boolean success = false;
        if (br.hasErrors()) {
            message = "输入的信息有错误！";
            writeJsonResponse(response, success, message);
            return;
        }
        String userPhotoFileName = this.handlePhotoUpload(request,  fileKeyName: "userPhotoFile");
        if(!userPhotoFileName.equals("upload/NoImage.jpg"))userInfo.setUserPhoto(userPhotoFileName);


        try {
            userInfoService.updateUserInfo(userInfo);
            message = "用户更新成功!";
            success = true;
            writeJsonResponse(response, success, message);
        } catch (Exception e) {
            e.printStackTrace();
            message = "用户更新失败!";
            writeJsonResponse(response, success, message);
        }
    }
    /*删除用户信息*/
    @RequestMapping(value="/{user_name}/delete",method=RequestMethod.GET)
    public String delete(@PathVariable String user_name,HttpServletRequest request) throws UnsupportedEncodingException {
          try {
              userInfoService.deleteUserInfo(user_name);
                request.setAttribute( s: "message",  o: "用户删除成功!");
                return "message";
            } catch (Exception e) {
                e.printStackTrace();
                request.setAttribute( s: "error",  o: "用户删除失败!");
                return "error";

            }

    }
```

```java
    /*ajax方式删除多条用户记录*/
    @RequestMapping(value="/deletes",method=RequestMethod.POST)
    public void delete(String user_names,HttpServletRequest request,HttpServletResponse response) throws IOException, JSONException {
        String message = "";
        boolean success = false;
        try {
            int count = userInfoService.deleteUserInfos(user_names);
            success = true;
            message = count + "条记录删除成功";
            writeJsonResponse(response, success, message);
        } catch (Exception e) {
            //e.printStackTrace();
            message = "有记录存在外键约束,删除失败";
            writeJsonResponse(response, success, message);
        }
    }

    /*按照查询条件导出用户信息到Excel*/
    @RequestMapping(value = { "/OutToExcel" }, method = {RequestMethod.GET,RequestMethod.POST})
    public void OutToExcel(String user_name,String name,String birthDate,String telephone, Model model, HttpServletRequest request,HttpServl
        if(user_name == null) user_name = "";
        if(name == null) name = "";
        if(birthDate == null) birthDate = "";
        if(telephone == null) telephone = "";
        List<UserInfo> userInfoList = userInfoService.queryUserInfo(user_name,name,birthDate,telephone);
        ExportExcelUtil ex = new ExportExcelUtil();
        String _title = "UserInfo信息记录";
        String[] headers = { "用户名","姓名","性别","出生日期","用户照片","联系电话","邮箱","注册时间"};
        List<String[]> dataset = new ArrayList<>();
        for(int i=0;i<userInfoList.size();i++) {
            UserInfo userInfo = userInfoList.get(i);
            dataset.add(new String[]{userInfo.getUser_name(),userInfo.getName(),userInfo.getGender(),userInfo.getBirthDate(),userInfo.getUse
        }
        /*
        OutputStream out = null;
        try {
            out = new FileOutputStream("C://output.xls");
```

```java
            ex.exportExcel(title,headers, dataset, out);
            out.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
        */
        OutputStream out = null;//创建一个输出流对象
        try {
            out = response.getOutputStream();//
            response.setHeader( s: "Content-disposition", s1: "attachment; filename="+"UserInfo.xls");//filename是下载的xls的名，建议最好用英文
            response.setContentType("application/msexcel;charset=UTF-8");//设置类型
            response.setHeader( s: "Pragma", s1: "No-cache");//设置头
            response.setHeader( s: "Cache-Control", s1: "no-cache");//设置头
            response.setDateHeader( s: "Expires", l: 0);//设置日期头
            String rootPath = request.getSession().getServletContext().getRealPath( s: "/");
            ex.exportExcel(rootPath,_title,headers, dataset, out);
            out.flush();
        } catch (IOException e) {
            e.printStackTrace();
        }finally{
            try{
                if(out!=null){
                    out.close();
                }
            }catch(IOException e){
                e.printStackTrace();
            }
        }
    }
}
```

mapper 包：

AdminMapper:

AdminMapper.java

```java
package com.chengxusheji.mapper;


import com.chengxusheji.po.Admin;

3 usages
public interface AdminMapper {

    2 usages
    public Admin findAdminByUserName(String username) throws Exception;

    1 usage
    public void changePassword(Admin admin) throws Exception;

    1 usage
    public void addAdmin(Admin admin) throws Exception;

}
```

DepartmentMapper:

DepartmentMapper.java

```java
package com.chengxusheji.mapper;

import java.util.ArrayList;
import org.apache.ibatis.annotations.Param;
import com.chengxusheji.po.Department;

4 usages
public interface DepartmentMapper {
    /*添加科室信息信息*/
    1 usage
    public void addDepartment(Department department) throws Exception;

    /*按照查询条件分页查询科室信息记录*/
    1 usage
    public ArrayList<Department> queryDepartment(@Param("where") String where,@Param("startIndex") int startIndex,@Param("pageSize") int pageSize)

    /*按照查询条件查询所有科室信息记录*/
    2 usages
    public ArrayList<Department> queryDepartmentList(@Param("where") String where) throws Exception;

    /*按照查询条件的科室信息记录数*/
    1 usage
    public int queryDepartmentCount(@Param("where") String where) throws Exception;

    /*根据主键查询某条科室信息记录*/
    1 usage
    public Department getDepartment(int departmentId) throws Exception;

    /*更新科室信息记录*/
    1 usage
    public void updateDepartment(Department department) throws Exception;

    /*删除科室信息记录*/
    2 usages
    public void deleteDepartment(int departmentId) throws Exception;

}
```

DoctorMapper:

DoctorMapper.java

```java
package com.chengxusheji.mapper;

import java.util.ArrayList;
import org.apache.ibatis.annotations.Param;
import com.chengxusheji.po.Doctor;

5 usages
public interface DoctorMapper {
    /*添加医生信息信息*/
    1 usage
    public void addDoctor(Doctor doctor) throws Exception;

    /*按照查询条件分页查询医生信息记录*/
    1 usage
    public ArrayList<Doctor> queryDoctor(@Param("where") String where,@Param("startIndex") int startIndex,@Param("pageSize") int pageSize) throw

    /*按照查询条件查询所有医生信息记录*/
    2 usages
    public ArrayList<Doctor> queryDoctorList(@Param("where") String where) throws Exception;

    /*按照查询条件的医生信息记录数*/
    1 usage
    public int queryDoctorCount(@Param("where") String where) throws Exception;

    /*根据主键查询某条医生信息记录*/
    1 usage
    public Doctor getDoctor(String doctorNumber) throws Exception;

    /*更新医生信息记录*/
    1 usage
    public void updateDoctor(Doctor doctor) throws Exception;

    /*删除医生信息记录*/
    2 usages
    public void deleteDoctor(String doctorNumber) throws Exception;

}
```

LeavewordMapper：

LeavewordMapper.java

```java
package com.chengxusheji.mapper;

import java.util.ArrayList;
import org.apache.ibatis.annotations.Param;
import com.chengxusheji.po.Leaveword;

3 usages
public interface LeavewordMapper {
    /*添加留言信息*/
    1 usage
    public void addLeaveword(Leaveword leaveword) throws Exception;

    /*按照查询条件分页查询留言记录*/
    1 usage
    public ArrayList<Leaveword> queryLeaveword(@Param("where") String where,@Param("startIndex") int startIndex,@Param("pageSize") int pageSize) th

    /*按照查询条件查询所有留言记录*/
    2 usages
    public ArrayList<Leaveword> queryLeavewordList(@Param("where") String where) throws Exception;

    /*按照查询条件的留言记录数*/
    1 usage
    public int queryLeavewordCount(@Param("where") String where) throws Exception;

    /*根据主键查询某条留言记录*/
    1 usage
    public Leaveword getLeaveword(int leaveWordId) throws Exception;

    /*更新留言记录*/
    1 usage
    public void updateLeaveword(Leaveword leaveword) throws Exception;

    /*删除留言记录*/
    2 usages
    public void deleteLeaveword(int leaveWordId) throws Exception;

}
```

NewsMapper：

NewsMapper.java

```java
package com.chengxusheji.mapper;

import java.util.ArrayList;
import org.apache.ibatis.annotations.Param;
import com.chengxusheji.po.News;

3 usages
public interface NewsMapper {
    /*添加新闻信息信息*/
    1 usage
    public void addNews(News news) throws Exception;

    /*按照查询条件分页查询新闻信息记录*/
    1 usage
    public ArrayList<News> queryNews(@Param("where") String where,@Param("startIndex") int startIndex,@Param("pageSize") int pageSize) throws Ex

    /*按照查询条件查询所有新闻信息记录*/
    2 usages
    public ArrayList<News> queryNewsList(@Param("where") String where) throws Exception;

    /*按照查询条件的新闻信息记录数*/
    1 usage
    public int queryNewsCount(@Param("where") String where) throws Exception;

    /*根据主键查询某条新闻信息记录*/
    1 usage
    public News getNews(int newsId) throws Exception;

    /*更新新闻信息记录*/
    1 usage
    public void updateNews(News news) throws Exception;

    /*删除新闻信息记录*/
    2 usages
    public void deleteNews(int newsId) throws Exception;

}
```

OrderInfoMapper：

OrderInfoMapper.java

```java
package com.chengxusheji.mapper;

import java.util.ArrayList;
import org.apache.ibatis.annotations.Param;
import com.chengxusheji.po.OrderInfo;

3 usages
public interface OrderInfoMapper {
    /*添加预约信息信息*/
    1 usage
    public void addOrderInfo(OrderInfo orderInfo) throws Exception;

    /*按照查询条件分页查询预约信息记录*/
    1 usage
    public ArrayList<OrderInfo> queryOrderInfo(@Param("where") String where,@Param("startIndex") int startIndex,@Param("pageSize") int pageSize) th

    /*按照查询条件查询所有预约信息记录*/
    2 usages
    public ArrayList<OrderInfo> queryOrderInfoList(@Param("where") String where) throws Exception;

    /*按照查询条件的预约信息记录数*/
    1 usage
    public int queryOrderInfoCount(@Param("where") String where) throws Exception;

    /*根据主键查询某条预约信息记录*/
    1 usage
    public OrderInfo getOrderInfo(int orderId) throws Exception;

    /*更新预约信息记录*/
    1 usage
    public void updateOrderInfo(OrderInfo orderInfo) throws Exception;

    /*删除预约信息记录*/
    2 usages
    public void deleteOrderInfo(int orderId) throws Exception;

}
```

PatientMapper：

PatientMapper.java

```java
package com.chengxusheji.mapper;

import java.util.ArrayList;
import org.apache.ibatis.annotations.Param;
import com.chengxusheji.po.Patient;

3 usages
public interface PatientMapper {
    /*添加病人信息信息*/
    1 usage
    public void addPatient(Patient patient) throws Exception;

    /*按照查询条件分页查询病人信息记录*/
    1 usage
    public ArrayList<Patient> queryPatient(@Param("where") String where,@Param("startIndex") int startIndex,@Param("pageSize") int pageSize) throw

    /*按照查询条件查询所有病人信息记录*/
    2 usages
    public ArrayList<Patient> queryPatientList(@Param("where") String where) throws Exception;

    /*按照查询条件的病人信息记录数*/
    1 usage
    public int queryPatientCount(@Param("where") String where) throws Exception;

    /*根据主键查询某条病人信息记录*/
    1 usage
    public Patient getPatient(int patientId) throws Exception;

    /*更新病人信息记录*/
    1 usage
    public void updatePatient(Patient patient) throws Exception;

    /*删除病人信息记录*/
    2 usages
    public void deletePatient(int patientId) throws Exception;

}
```

UserInfoMapper：

UserInfoMapper.java

```java
package com.chengxusheji.mapper;

import java.util.ArrayList;
import org.apache.ibatis.annotations.Param;
import com.chengxusheji.po.UserInfo;

5 usages
public interface UserInfoMapper {
    /*添加用户信息*/
    1 usage
    public void addUserInfo(UserInfo userInfo) throws Exception;

    /*按照查询条件分页查询用户记录*/
    1 usage
    public ArrayList<UserInfo> queryUserInfo(@Param("where") String where,@Param("startIndex") int startIndex,@Param("pageSize") int pageSize) thr

    /*按照查询条件查询所有用户记录*/
    2 usages
    public ArrayList<UserInfo> queryUserInfoList(@Param("where") String where) throws Exception;

    /*按照查询条件的用户记录数*/
    1 usage
    public int queryUserInfoCount(@Param("where") String where) throws Exception;

    /*根据主键查询某条用户记录*/
    1 usage
    public UserInfo getUserInfo(String user_name) throws Exception;

    /*更新用户记录*/
    1 usage
    public void updateUserInfo(UserInfo userInfo) throws Exception;

    /*删除用户记录*/
    2 usages
    public void deleteUserInfo(String user_name) throws Exception;

}
```

po 包：

Admin：

Admin.java

```java
import org.hibernate.validator.constraints.NotEmpty;


28 usages
public class Admin {

    2 usages
    private Integer id;

    /**
     * 类型 0 管理员 1 医生 2 患者
     */
    2 usages
    private Integer type;
    /*管理员用户名*/
    2 usages
    @NotEmpty(message="用户名不能为空")
    private String username;
    /*登陆密码*/
    2 usages
    @NotEmpty(message="登陆密码不能为空")
    private String password;

    public Integer getId() { return id; }

    public void setId(Integer id) { this.id = id; }

    public Integer getType() { return type; }

    public void setType(Integer type) { this.type = type; }

    public String getUsername() { return username; }
    public void setUsername(String username) { this.username = username; }
    public String getPassword() { return password; }
    public void setPassword(String password) { this.password = password; }

}
```

Department：

Department.java

```java
package com.chengxusheji.po;

import javax.validation.constraints.NotNull;
import org.hibernate.validator.constraints.NotEmpty;
import org.json.JSONException;
import org.json.JSONObject;

public class Department {
    /*科室id*/
    2 usages
    private Integer departmentId;
    public Integer getDepartmentId() { return departmentId; }
    public void setDepartmentId(Integer departmentId) { this.departmentId = departmentId; }

    /*科室名称*/
    2 usages
    @NotEmpty(message="科室名称不能为空")
    private String departmentName;
    public String getDepartmentName() { return departmentName; }
    public void setDepartmentName(String departmentName) { this.departmentName = departmentName; }

    /*科室介绍*/
    2 usages
    private String departmentDesc;
    public String getDepartmentDesc() { return departmentDesc; }
    public void setDepartmentDesc(String departmentDesc) { this.departmentDesc = departmentDesc; }

    /*成立日期*/
    2 usages
    @NotEmpty(message="成立日期不能为空")
    private String birthDate;
    public String getBirthDate() { return birthDate; }
    public void setBirthDate(String birthDate) { this.birthDate = birthDate; }

    /*负责人*/
    2 usages
    private String chargeMan;
    public String getChargeMan() { return chargeMan; }
    public void setChargeMan(String chargeMan) { this.chargeMan = chargeMan; }

    public JSONObject getJsonObject() throws JSONException {
        JSONObject jsonDepartment=new JSONObject();
        jsonDepartment.accumulate("departmentId", this.getDepartmentId());
        jsonDepartment.accumulate("departmentName", this.getDepartmentName());
        jsonDepartment.accumulate("departmentDesc", this.getDepartmentDesc());
        jsonDepartment.accumulate("birthDate", this.getBirthDate().length()>19?this.getBirthDate().substring(0,19):this.getBirthDate());
        jsonDepartment.accumulate("chargeMan", this.getChargeMan());
        return jsonDepartment;
    }}
```

Doctor：

Doctor.java

```java
package com.chengxusheji.po;

import javax.validation.constraints.NotNull;
import org.hibernate.validator.constraints.NotEmpty;
import org.json.JSONException;
import org.json.JSONObject;

public class Doctor {

    2 usages
    private Integer id;
    2 usages
    private Integer adminid;
    /*医生工号*/
    2 usages
    @NotEmpty(message="医生工号不能为空")
    private String doctorNumber;
    public String getDoctorNumber() { return doctorNumber; }
    public void setDoctorNumber(String doctorNumber) { this.doctorNumber = doctorNumber; }

    /*登录密码*/
    2 usages
    private String password;
    public String getPassword() { return password; }
    public void setPassword(String password) { this.password = password; }

    /*所在科室*/
    2 usages
    private Department departmentObj;
    public Department getDepartmentObj() { return departmentObj; }
    public void setDepartmentObj(Department departmentObj) { this.departmentObj = departmentObj; }

    /*医生姓名*/
    2 usages
    @NotEmpty(message="医生姓名不能为空")
    private String doctorName;
    public String getDoctorName() { return doctorName; }
    public void setDoctorName(String doctorName) { this.doctorName = doctorName; }

    /*性别*/
    2 usages
    @NotEmpty(message="性别不能为空")
    private String sex;
    public String getSex() { return sex; }
    public void setSex(String sex) { this.sex = sex; }

    /*医生照片*/
    2 usages
    private String doctorPhoto;
    4 usages
    public String getDoctorPhoto() { return doctorPhoto; }
    2 usages
    public void setDoctorPhoto(String doctorPhoto) { this.doctorPhoto = doctorPhoto; }

    /*出生日期*/
    2 usages
    @NotEmpty(message="出生日期不能为空")
    private String birthDate;
    public String getBirthDate() { return birthDate; }
    public void setBirthDate(String birthDate) { this.birthDate = birthDate; }

    /*医生职位*/
    2 usages
    @NotEmpty(message="医生职位不能为空")
    private String position;
    public String getPosition() { return position; }
    public void setPosition(String position) { this.position = position; }

    /*工作经验*/
    2 usages
    @NotEmpty(message="工作经验不能为空")
    private String experience;
    public String getExperience() { return experience; }
    public void setExperience(String experience) { this.experience = experience; }

}
```

```
    /*联系方式*/
    2 usages
    private String contactWay;
    public String getContactWay() { return contactWay; }
    public void setContactWay(String contactWay) { this.contactWay = contactWay; }

    /*擅长*/
    2 usages
    private String goodAt;
    public String getGoodAt() { return goodAt; }
    public void setGoodAt(String goodAt) { this.goodAt = goodAt; }

    /*医生介绍*/
    2 usages
    @NotEmpty(message="医生介绍不能为空")
    private String doctorDesc;
    public String getDoctorDesc() { return doctorDesc; }
    public void setDoctorDesc(String doctorDesc) { this.doctorDesc = doctorDesc; }

    public Integer getId() { return id; }

    public void setId(Integer id) { this.id = id; }

    no usages
    public Integer getAdminid() { return adminid; }

    1 usage
    public void setAdminid(Integer adminid) { this.adminid = adminid; }

    public JSONObject getJsonObject() throws JSONException {
        JSONObject jsonDoctor=new JSONObject();
        jsonDoctor.accumulate("doctorNumber", this.getDoctorNumber());
        jsonDoctor.accumulate("password", this.getPassword());
        jsonDoctor.accumulate("departmentObj", this.getDepartmentObj().getDepartmentName());
        jsonDoctor.accumulate("departmentObjPri", this.getDepartmentObj().getDepartmentId());
        jsonDoctor.accumulate("doctorName", this.getDoctorName());
        jsonDoctor.accumulate("sex", this.getSex());
        jsonDoctor.accumulate("doctorPhoto", this.getDoctorPhoto());
        jsonDoctor.accumulate("birthDate", this.getBirthDate().length()>19?this.getBirthDate().substring(0,19):this.getBirthDate());
        jsonDoctor.accumulate("position", this.getPosition());
        jsonDoctor.accumulate("experience", this.getExperience());
        jsonDoctor.accumulate("contactWay", this.getContactWay());
        jsonDoctor.accumulate("goodAt", this.getGoodAt());
        jsonDoctor.accumulate("doctorDesc", this.getDoctorDesc());
        return jsonDoctor;
    }}
```

Leaveword：

Leaveword.java

```java
package com.chengxusheji.po;

import javax.validation.constraints.NotNull;
import org.hibernate.validator.constraints.NotEmpty;
import org.json.JSONException;
import org.json.JSONObject;

public class Leaveword {
    /*留言id*/
    2 usages
    private Integer leaveWordId;
    public Integer getLeaveWordId() { return leaveWordId; }
    public void setLeaveWordId(Integer leaveWordId) { this.leaveWordId = leaveWordId; }

    /*留言标题*/
    2 usages
    @NotEmpty(message="留言标题不能为空")
    private String leaveTitle;
    public String getLeaveTitle() { return leaveTitle; }
    public void setLeaveTitle(String leaveTitle) { this.leaveTitle = leaveTitle; }

    /*留言内容*/
    2 usages
    @NotEmpty(message="留言内容不能为空")
    private String leaveContent;
    public String getLeaveContent() { return leaveContent; }
    public void setLeaveContent(String leaveContent) { this.leaveContent = leaveContent; }

    /*留言人*/
    2 usages
    private UserInfo userObj;
    public UserInfo getUserObj() { return userObj; }
    public void setUserObj(UserInfo userObj) { this.userObj = userObj; }

    /*留言时间*/
    2 usages
    private String leaveTime;
    public String getLeaveTime() { return leaveTime; }
    public void setLeaveTime(String leaveTime) { this.leaveTime = leaveTime; }

    /*管理回复*/
    2 usages
    private String replyContent;
    public String getReplyContent() { return replyContent; }
    public void setReplyContent(String replyContent) { this.replyContent = replyContent; }

    /*回复时间*/
    2 usages
    private String replyTime;
    public String getReplyTime() { return replyTime; }
    public void setReplyTime(String replyTime) { this.replyTime = replyTime; }

    public JSONObject getJsonObject() throws JSONException {
        JSONObject jsonLeaveword=new JSONObject();
        jsonLeaveword.accumulate("leaveWordId", this.getLeaveWordId());
        jsonLeaveword.accumulate("leaveTitle", this.getLeaveTitle());
        jsonLeaveword.accumulate("leaveContent", this.getLeaveContent());
        jsonLeaveword.accumulate("userObj", this.getUserObj().getName());
        jsonLeaveword.accumulate("userObjPri", this.getUserObj().getUser_name());
        jsonLeaveword.accumulate("leaveTime", this.getLeaveTime().length()>19?this.getLeaveTime().substring(0,19):this.getLeaveTime());
        jsonLeaveword.accumulate("replyContent", this.getReplyContent());
        jsonLeaveword.accumulate("replyTime", this.getReplyTime()!=null?(this.getReplyTime().length()>19?this.getReplyTime().substring(0,19):this.g
        return jsonLeaveword;
    }}
```

News：

News.java

```java
package com.chengxusheji.po;

import javax.validation.constraints.NotNull;
import org.hibernate.validator.constraints.NotEmpty;
import org.json.JSONException;
import org.json.JSONObject;

public class News {
    /*新闻id*/
    2 usages
    private Integer newsId;
    public Integer getNewsId() { return newsId; }
    public void setNewsId(Integer newsId) { this.newsId = newsId; }

    /*新闻标题*/
    2 usages
    @NotEmpty(message="新闻标题不能为空")
    private String newsTitle;
    public String getNewsTitle() { return newsTitle; }
    public void setNewsTitle(String newsTitle) { this.newsTitle = newsTitle; }

    /*新闻图片*/
    2 usages
    private String newsPhoto;
    4 usages
    public String getNewsPhoto() { return newsPhoto; }
    2 usages
    public void setNewsPhoto(String newsPhoto) { this.newsPhoto = newsPhoto; }

    /*新闻内容*/
    2 usages
    @NotEmpty(message="新闻内容不能为空")
    private String newsContent;
    public String getNewsContent() { return newsContent; }
    public void setNewsContent(String newsContent) { this.newsContent = newsContent; }

    /*新闻日期*/
    2 usages
    @NotEmpty(message="新闻日期不能为空")
    private String newsDate;
    public String getNewsDate() { return newsDate; }
    public void setNewsDate(String newsDate) { this.newsDate = newsDate; }

    /*新闻来源*/
    2 usages
    private String newsFrom;
    public String getNewsFrom() { return newsFrom; }
    public void setNewsFrom(String newsFrom) { this.newsFrom = newsFrom; }

    public JSONObject getJsonObject() throws JSONException {
        JSONObject jsonNews=new JSONObject();
        jsonNews.accumulate("newsId", this.getNewsId());
        jsonNews.accumulate("newsTitle", this.getNewsTitle());
        jsonNews.accumulate("newsPhoto", this.getNewsPhoto());
        jsonNews.accumulate("newsContent", this.getNewsContent());
        jsonNews.accumulate("newsDate", this.getNewsDate().length()>19?this.getNewsDate().substring(0,19):this.getNewsDate());
        jsonNews.accumulate("newsFrom", this.getNewsFrom());
        return jsonNews;
    }}
```

OrderInfo：

OrderInfo.java

```java
package com.chengxusheji.po;

import javax.validation.constraints.NotNull;
import org.hibernate.validator.constraints.NotEmpty;
import org.json.JSONException;
import org.json.JSONObject;

public class OrderInfo {
    /*预约id*/
    2 usages
    private Integer orderId;
    public Integer getOrderId() { return orderId; }
    public void setOrderId(Integer orderId) { this.orderId = orderId; }

    /*预约用户*/
    2 usages
    private UserInfo userObj;
    public UserInfo getUserObj() { return userObj; }
    public void setUserObj(UserInfo userObj) { this.userObj = userObj; }

    /*预约医生*/
    2 usages
    private Doctor doctorObj;
    public Doctor getDoctorObj() { return doctorObj; }
    public void setDoctorObj(Doctor doctorObj) { this.doctorObj = doctorObj; }

    /*预约日期*/
    2 usages
    @NotEmpty(message="预约日期不能为空")
    private String orderDate;
    public String getOrderDate() { return orderDate; }
    public void setOrderDate(String orderDate) { this.orderDate = orderDate; }

    /*时段*/
    2 usages
    @NotEmpty(message="时段不能为空")
    private String timeInterval;
    public String getTimeInterval() { return timeInterval; }
    public void setTimeInterval(String timeInterval) { this.timeInterval = timeInterval; }

    /*联系电话*/
    2 usages
    @NotEmpty(message="联系电话不能为空")
    private String telephone;
    public String getTelephone() { return telephone; }
    public void setTelephone(String telephone) { this.telephone = telephone; }

    /*下单时间*/
    2 usages
    @NotEmpty(message="下单时间不能为空")
    private String orderTime;
    public String getOrderTime() { return orderTime; }
    public void setOrderTime(String orderTime) { this.orderTime = orderTime; }

    /*处理状态*/
    2 usages
    @NotEmpty(message="处理状态不能为空")
    private String checkState;
    public String getCheckState() { return checkState; }
    public void setCheckState(String checkState) { this.checkState = checkState; }

    /*医生回复*/
    2 usages
    @NotEmpty(message="医生回复不能为空")
    private String replyContent;
    public String getReplyContent() { return replyContent; }
    public void setReplyContent(String replyContent) { this.replyContent = replyContent; }

    public JSONObject getJsonObject() throws JSONException {
        JSONObject jsonOrderInfo=new JSONObject();
        jsonOrderInfo.accumulate("orderId", this.getOrderId());
```

```
        jsonOrderInfo.accumulate("userObj", this.getUserObj().getName());
        jsonOrderInfo.accumulate("userObjPri", this.getUserObj().getUser_name());
        jsonOrderInfo.accumulate("doctorObj", this.getDoctorObj().getDoctorName());
        jsonOrderInfo.accumulate("doctorObjPri", this.getDoctorObj().getDoctorNumber());
        jsonOrderInfo.accumulate("orderDate", this.getOrderDate().length()>19?this.getOrderDate().substring(0,19):this.getOrderDate());
        jsonOrderInfo.accumulate("timeInterval", this.getTimeInterval());
        jsonOrderInfo.accumulate("telephone", this.getTelephone());
        jsonOrderInfo.accumulate("orderTime", this.getOrderTime().length()>19?this.getOrderTime().substring(0,19):this.getOrderTime());
        jsonOrderInfo.accumulate("checkState", this.getCheckState());
        jsonOrderInfo.accumulate("replyContent", this.getReplyContent());
        return jsonOrderInfo;
    }}
```

Patient：

Patient.java

```
package com.chengxusheji.po;

import javax.validation.constraints.NotNull;
import org.hibernate.validator.constraints.NotEmpty;
import org.json.JSONException;
import org.json.JSONObject;

public class Patient {
    /*病人id*/
    2 usages
    private Integer patientId;
    public Integer getPatientId() { return patientId; }
    public void setPatientId(Integer patientId) { this.patientId = patientId; }

    /*医生*/
    2 usages
    private Doctor doctorObj;
    public Doctor getDoctorObj() { return doctorObj; }
    public void setDoctorObj(Doctor doctorObj) { this.doctorObj = doctorObj; }

    /*病人姓名*/
    2 usages
    @NotEmpty(message="病人姓名不能为空")
    private String patientName;
    public String getPatientName() { return patientName; }
    public void setPatientName(String patientName) { this.patientName = patientName; }

    /*性别*/
    2 usages
    @NotEmpty(message="性别不能为空")
    private String sex;
    public String getSex() { return sex; }
    public void setSex(String sex) { this.sex = sex; }

    /*身份证号*/
    2 usages
    @NotEmpty(message="身份证号不能为空")
    private String cardNumber;
```

```
    public String getCardNumber() { return cardNumber; }
    public void setCardNumber(String cardNumber) { this.cardNumber = cardNumber; }

    /*联系电话*/
    2 usages
    @NotEmpty(message="联系电话不能为空")
    private String telephone;
    public String getTelephone() { return telephone; }
    public void setTelephone(String telephone) { this.telephone = telephone; }

    /*病人病例*/
    2 usages
    @NotEmpty(message="病人病例不能为空")
    private String illnessCase;
    public String getIllnessCase() { return illnessCase; }
    public void setIllnessCase(String illnessCase) { this.illnessCase = illnessCase; }

    /*登记时间*/
    2 usages
    @NotEmpty(message="登记时间不能为空")
    private String addTime;
    public String getAddTime() { return addTime; }
    public void setAddTime(String addTime) { this.addTime = addTime; }

    public JSONObject getJsonObject() throws JSONException {
        JSONObject jsonPatient=new JSONObject();
        jsonPatient.accumulate("patientId", this.getPatientId());
        jsonPatient.accumulate("doctorObj", this.getDoctorObj().getDoctorName());
        jsonPatient.accumulate("doctorObjPri", this.getDoctorObj().getDoctorNumber());
        jsonPatient.accumulate("patientName", this.getPatientName());
        jsonPatient.accumulate("sex", this.getSex());
        jsonPatient.accumulate("cardNumber", this.getCardNumber());
        jsonPatient.accumulate("telephone", this.getTelephone());
        jsonPatient.accumulate("illnessCase", this.getIllnessCase());
        jsonPatient.accumulate("addTime", this.getAddTime().length()>19?this.getAddTime().substring(0,19):this.getAddTime());
        return jsonPatient;
    }}
```

UserInfo：

UserInfo.java

```java
package com.chengxusheji.po;

import org.hibernate.validator.constraints.NotEmpty;
import org.json.JSONException;
import org.json.JSONObject;

public class UserInfo {

    2 usages
    private Integer id;
    2 usages
    private Integer adminid;
    /*用户名*/
    2 usages
    @NotEmpty(message="用户名不能为空")
    private String user_name;
    public String getUser_name() { return user_name; }
    public void setUser_name(String user_name) { this.user_name = user_name; }

    /*登录密码*/
    2 usages
    @NotEmpty(message="登录密码不能为空")
    private String password;
    public String getPassword() { return password; }
    public void setPassword(String password) { this.password = password; }

    /*姓名*/
    2 usages
    @NotEmpty(message="姓名不能为空")
    private String name;
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }

    /*性别*/
    2 usages
    @NotEmpty(message="性别不能为空")
    private String gender;
    public String getGender() { return gender; }
    public void setGender(String gender) { this.gender = gender; }

    /*出生日期*/
    2 usages
    @NotEmpty(message="出生日期不能为空")
    private String birthDate;
    public String getBirthDate() { return birthDate; }
    public void setBirthDate(String birthDate) { this.birthDate = birthDate; }

    /*用户照片*/
    2 usages
    private String userPhoto;
    public String getUserPhoto() { return userPhoto; }
    public void setUserPhoto(String userPhoto) { this.userPhoto = userPhoto; }

    /*联系电话*/
    2 usages
    @NotEmpty(message="联系电话不能为空")
    private String telephone;
    public String getTelephone() { return telephone; }
    public void setTelephone(String telephone) { this.telephone = telephone; }

    /*邮箱*/
    2 usages
    @NotEmpty(message="邮箱不能为空")
    private String email;
    public String getEmail() { return email; }
    public void setEmail(String email) { this.email = email; }

    /*家庭地址*/
    2 usages
    private String address;
    public String getAddress() { return address; }
    public void setAddress(String address) { this.address = address; }

}
```

```
    /*注册时间*/
    2 usages
    private String regTime;
    public String getRegTime() { return regTime; }
    public void setRegTime(String regTime) { this.regTime = regTime; }
    public Integer getId() { return id; }

    public void setId(Integer id) { this.id = id; }

    1 usage
    public Integer getAdminid() { return adminid; }

    1 usage
    public void setAdminid(Integer adminid) { this.adminid = adminid; }
    public JSONObject getJsonObject() throws JSONException {
        JSONObject jsonUserInfo=new JSONObject();
        jsonUserInfo.accumulate("id", this.getId());
        jsonUserInfo.accumulate("adminid", this.getAdminid());
        jsonUserInfo.accumulate("user_name", this.getUser_name());
        jsonUserInfo.accumulate("password", this.getPassword());
        jsonUserInfo.accumulate("name", this.getName());
        jsonUserInfo.accumulate("gender", this.getGender());
        jsonUserInfo.accumulate("birthDate", this.getBirthDate().length()>19?this.getBirthDate().substring(0,19):this.getBirthDate());
        jsonUserInfo.accumulate("userPhoto", this.getUserPhoto());
        jsonUserInfo.accumulate("telephone", this.getTelephone());
        jsonUserInfo.accumulate("email", this.getEmail());
        jsonUserInfo.accumulate("address", this.getAddress());
        jsonUserInfo.accumulate("regTime", this.getRegTime().length()>19?this.getRegTime().substring(0,19):this.getRegTime());
        return jsonUserInfo;
    }}
```

Service 包：

AdminService：

AdminService.java

```
package com.chengxusheji.service;

import com.chengxusheji.po.UserInfo;
import javax.annotation.Resource;


import org.springframework.stereotype.Service;

import com.chengxusheji.mapper.AdminMapper;
import com.chengxusheji.po.Admin;

6 usages
@Service
public class AdminService {
    4 usages
    @Resource AdminMapper adminMapper;

    /*保存业务逻辑错误信息字段*/
    5 usages
    private String errMessage;
    1 usage
    public String getErrMessage() { return this.errMessage; }

    /*验证用户登录*/
    2 usages
    public Admin checkLogin(Admin admin) throws Exception {
        Admin db_admin = (Admin) adminMapper.findAdminByUserName(admin.getUsername());
        if(db_admin == null) {
            this.errMessage = " 账号不存在 ";
            System.out.print(this.errMessage);
            return null;
        } else if( !db_admin.getPassword().equals(admin.getPassword())) {
            this.errMessage = " 密码不正确！ ";
            System.out.print(this.errMessage);
            return null;
        }

```

```
        return db_admin;
    }

    /*添加用户记录*/
    3 usages
    public void addAdmin(Admin admin) throws Exception {
        adminMapper.addAdmin(admin);
    }


    /*修改用户登录密码*/
    1 usage
    public void changePassword(String username, String newPassword) throws Exception {
        Admin admin = new Admin();
        admin.setUsername(username);
        admin.setPassword(newPassword);
        adminMapper.changePassword(admin);
    }

    /*根据用户名获取管理员对象*/
    6 usages
    public Admin findAdminByUserName(String username) throws Exception {
        Admin db_admin = null;
        db_admin = adminMapper.findAdminByUserName(username);
        return db_admin;
    }
}
```

DepartmentSeivice：

DepartmentService.java

```
package com.chengxusheji.service;

import java.util.ArrayList;
import javax.annotation.Resource;
import org.springframework.stereotype.Service;
import com.chengxusheji.po.Department;

import com.chengxusheji.mapper.DepartmentMapper;
4 usages
@Service
public class DepartmentService {

    9 usages
    @Resource DepartmentMapper departmentMapper;
    /*每页显示记录数目*/
    6 usages
    private int rows = 10;;
    public int getRows() { return rows; }
    public void setRows(int rows) { this.rows = rows; }

    /*保存查询后总的页数*/
    4 usages
    private int totalPage;
    public void setTotalPage(int totalPage) { this.totalPage = totalPage; }
    public int getTotalPage() { return totalPage; }

    /*保存查询到的总记录数*/
    5 usages
    private int recordNumber;
    public void setRecordNumber(int recordNumber) { this.recordNumber = recordNumber; }
    public int getRecordNumber() { return recordNumber; }

    /*添加科室信息记录*/
    1 usage
    public void addDepartment(Department department) throws Exception {
        departmentMapper.addDepartment(department);
    }
```

```
    /*按照查询条件分页查询科室信息记录*/
    2 usages
    public ArrayList<Department> queryDepartment(String departmentName,String birthDate,int currentPage) throws Exception {
        String where = "where 1=1";
        if(!departmentName.equals("")) where = where + " and t_department.departmentName like '%" + departmentName + "%'";
        if(!birthDate.equals("")) where = where + " and t_department.birthDate like '%" + birthDate + "%'";
        int startIndex = (currentPage-1) * this.rows;
        return departmentMapper.queryDepartment(where, startIndex, this.rows);
    }

    /*按照查询条件查询所有记录*/
    1 usage
    public ArrayList<Department> queryDepartment(String departmentName,String birthDate) throws Exception  {
        String where = "where 1=1";
        if(!departmentName.equals("")) where = where + " and t_department.departmentName like '%" + departmentName + "%'";
        if(!birthDate.equals("")) where = where + " and t_department.birthDate like '%" + birthDate + "%'";
        return departmentMapper.queryDepartmentList(where);
    }

    /*查询所有科室信息记录*/
    4 usages
    public ArrayList<Department> queryAllDepartment()  throws Exception {
        return departmentMapper.queryDepartmentList( where: "where 1=1");
    }

    /*当前查询条件下计算总的页数和记录数*/
    public void queryTotalPageAndRecordNumber(String departmentName,String birthDate) throws Exception {
        String where = "where 1=1";
        if(!departmentName.equals("")) where = where + " and t_department.departmentName like '%" + departmentName + "%'";
        if(!birthDate.equals("")) where = where + " and t_department.birthDate like '%" + birthDate + "%'";
        recordNumber = departmentMapper.queryDepartmentCount(where);
        int mod = recordNumber % this.rows;
        totalPage = recordNumber / this.rows;
        if(mod != 0) totalPage++;
```

```
    }

    /*根据主键获取科室信息记录*/
    2 usages
    public Department getDepartment(int departmentId) throws Exception  {
        Department department = departmentMapper.getDepartment(departmentId);
        return department;
    }

    /*更新科室信息记录*/
    1 usage
    public void updateDepartment(Department department) throws Exception {
        departmentMapper.updateDepartment(department);
    }

    /*删除一条科室信息记录*/
    1 usage
    public void deleteDepartment (int departmentId) throws Exception {
        departmentMapper.deleteDepartment(departmentId);
    }

    /*删除多条科室信息信息*/
    1 usage
    public int deleteDepartments (String departmentIds) throws Exception {
        String _departmentIds[] = departmentIds.split( regex: ",");
        for(String _departmentId: _departmentIds) {
            departmentMapper.deleteDepartment(Integer.parseInt(_departmentId));
        }
        return _departmentIds.length;
    }
}
```

DoctorService：

DoctorService.java

```java
package com.chengxusheji.service;

import java.util.ArrayList;
import javax.annotation.Resource;
import org.springframework.stereotype.Service;
import com.chengxusheji.po.Department;
import com.chengxusheji.po.Doctor;

import com.chengxusheji.mapper.DoctorMapper;
6 usages
@Service
public class DoctorService {

    9 usages
    @Resource DoctorMapper doctorMapper;
    /*每页显示记录数目*/
    6 usages
    private int rows = 10;;
    public int getRows() { return rows; }
    public void setRows(int rows) { this.rows = rows; }

    /*保存查询后总的页数*/
    4 usages
    private int totalPage;
    public void setTotalPage(int totalPage) { this.totalPage = totalPage; }
    public int getTotalPage() { return totalPage; }

    /*保存查询到的总记录数*/
    5 usages
    private int recordNumber;
    public void setRecordNumber(int recordNumber) { this.recordNumber = recordNumber; }
    public int getRecordNumber() { return recordNumber; }

    /*添加医生信息记录*/
    1 usage
    public void addDoctor(Doctor doctor) throws Exception {
        doctorMapper.addDoctor(doctor);
    }

    /*按照查询条件分页查询医生信息记录*/
    2 usages
    public ArrayList<Doctor> queryDoctor(String doctorNumber,Department departmentObj,String doctorName,String birthDate,int currentPage) throws Excep
        String where = "where 1=1";
        if(!doctorNumber.equals("")) where = where + " and t_doctor.doctorNumber like '%" + doctorNumber + "%'";
        if(null != departmentObj && departmentObj.getDepartmentId()!= null && departmentObj.getDepartmentId()!= 0)  where += " and t_doctor.department
        if(!doctorName.equals("")) where = where + " and t_doctor.doctorName like '%" + doctorName + "%'";
        if(!birthDate.equals("")) where = where + " and t_doctor.birthDate like '%" + birthDate + "%'";
        int startIndex = (currentPage-1) * this.rows;
        return doctorMapper.queryDoctor(where, startIndex, this.rows);
    }

    /*按照查询条件查询所有记录*/
    1 usage
    public ArrayList<Doctor> queryDoctor(String doctorNumber,Department departmentObj,String doctorName,String birthDate) throws Exception  {
        String where = "where 1=1";
        if(!doctorNumber.equals("")) where = where + " and t_doctor.doctorNumber like '%" + doctorNumber + "%'";
        if(null != departmentObj && departmentObj.getDepartmentId()!= null && departmentObj.getDepartmentId()!= 0)  where += " and t_doctor.department
        if(!doctorName.equals("")) where = where + " and t_doctor.doctorName like '%" + doctorName + "%'";
        if(!birthDate.equals("")) where = where + " and t_doctor.birthDate like '%" + birthDate + "%'";
        return doctorMapper.queryDoctorList(where);
    }

    /*查询所有医生信息记录*/
    7 usages
    public ArrayList<Doctor> queryAllDoctor()  throws Exception {
        return doctorMapper.queryDoctorList( where: "where 1=1");
    }

    /*当前查询条件下计算总的页数和记录数*/
    public void queryTotalPageAndRecordNumber(String doctorNumber,Department departmentObj,String doctorName,String birthDate) throws Exception {
        String where = "where 1=1";
        if(!doctorNumber.equals("")) where = where + " and t_doctor.doctorNumber like '%" + doctorNumber + "%'";
        if(null != departmentObj && departmentObj.getDepartmentId()!= null && departmentObj.getDepartmentId()!= 0)  where += " and t_doctor.department
        if(!doctorName.equals("")) where = where + " and t_doctor.doctorName like '%" + doctorName + "%'";
```

```
78          if(!birthDate.equals("")) where = where + " and t_doctor.birthDate like '%" + birthDate + "%'";
79          recordNumber = doctorMapper.queryDoctorCount(where);
80          int mod = recordNumber % this.rows;
81          totalPage = recordNumber / this.rows;
82          if(mod != 0) totalPage++;
83      }
84
85      /*根据主键获取医生信息记录*/
            3 usages
86      public Doctor getDoctor(String doctorNumber) throws Exception  {
87          Doctor doctor = doctorMapper.getDoctor(doctorNumber);
88          return doctor;
89      }
90
91      /*更新医生信息记录*/
            1 usage
92      public void updateDoctor(Doctor doctor) throws Exception {
93          doctorMapper.updateDoctor(doctor);
94      }
95
96      /*删除一条医生信息记录*/
            1 usage
97      public void deleteDoctor (String doctorNumber) throws Exception {
98          doctorMapper.deleteDoctor(doctorNumber);
99      }
100
101     /*删除多条医生信息信息*/
            1 usage
102 @   public int deleteDoctors (String doctorNumbers) throws Exception {
103         String _doctorNumbers[] = doctorNumbers.split( regex: ",");
104         for(String _doctorNumber: _doctorNumbers) {
105             doctorMapper.deleteDoctor(_doctorNumber);
106         }
107         return _doctorNumbers.length;
108     }
```

LeavewordService：

LeavewordService.java

```java
package com.chengxusheji.service;

import java.util.ArrayList;
import javax.annotation.Resource;
import org.springframework.stereotype.Service;
import com.chengxusheji.po.UserInfo;
import com.chengxusheji.po.Leaveword;

import com.chengxusheji.mapper.LeavewordMapper;
2 usages
@Service
public class LeavewordService {

    9 usages
    @Resource LeavewordMapper leavewordMapper;
    /*每页显示记录数目*/
    6 usages
    private int rows = 10;;
    public int getRows() { return rows; }
    public void setRows(int rows) { this.rows = rows; }

    /*保存查询后总的页数*/
    4 usages
    private int totalPage;
    public void setTotalPage(int totalPage) { this.totalPage = totalPage; }
    public int getTotalPage() { return totalPage; }

    /*保存查询到的总记录数*/
    5 usages
    private int recordNumber;
    public void setRecordNumber(int recordNumber) { this.recordNumber = recordNumber; }
    public int getRecordNumber() { return recordNumber; }

    /*添加留言记录*/
    1 usage
    public void addLeaveword(Leaveword leaveword) throws Exception {
        leavewordMapper.addLeaveword(leaveword);
    }

    /*按照查询条件分页查询留言记录*/
    2 usages
    public ArrayList<Leaveword> queryLeaveword(String leaveTitle,UserInfo userObj,String leaveTime,int currentPage) throws Exception {
        String where = "where 1=1";
        if(!leaveTitle.equals("")) where = where + " and t_leaveword.leaveTitle like '%" + leaveTitle + "%'";
        if(null != userObj &&  userObj.getUser_name() != null  && !userObj.getUser_name().equals(""))  where += " and t_leaveword.userObj='" + userObj
        if(!leaveTime.equals("")) where = where + " and t_leaveword.leaveTime like '%" + leaveTime + "%'";
        int startIndex = (currentPage-1) * this.rows;
        return leavewordMapper.queryLeaveword(where, startIndex, this.rows);
    }

    /*按照查询条件查询所有记录*/
    1 usage
    public ArrayList<Leaveword> queryLeaveword(String leaveTitle,UserInfo userObj,String leaveTime) throws Exception  {
        String where = "where 1=1";
        if(!leaveTitle.equals("")) where = where + " and t_leaveword.leaveTitle like '%" + leaveTitle + "%'";
        if(null != userObj &&  userObj.getUser_name() != null && !userObj.getUser_name().equals(""))  where += " and t_leaveword.userObj='" + userObj
        if(!leaveTime.equals("")) where = where + " and t_leaveword.leaveTime like '%" + leaveTime + "%'";
        return leavewordMapper.queryLeavewordList(where);
    }

    /*查询所有留言记录*/
    1 usage
    public ArrayList<Leaveword> queryAllLeaveword()  throws Exception {
        return leavewordMapper.queryLeavewordList( where: "where 1=1");
    }

    /*当前查询条件下计算总的页数和记录数*/
    public void queryTotalPageAndRecordNumber(String leaveTitle,UserInfo userObj,String leaveTime) throws Exception {
        String where = "where 1=1";
        if(!leaveTitle.equals("")) where = where + " and t_leaveword.leaveTitle like '%" + leaveTitle + "%'";
        if(null != userObj &&  userObj.getUser_name() != null && !userObj.getUser_name().equals(""))  where += " and t_leaveword.userObj='" + userObj.
        if(!leaveTime.equals("")) where = where + " and t_leaveword.leaveTime like '%" + leaveTime + "%'";
        recordNumber = leavewordMapper.queryLeavewordCount(where);
        int mod = recordNumber % this.rows;
```

```
78          totalPage = recordNumber / this.rows;
79          if(mod != 0) totalPage++;
80      }
81
82      /*根据主键获取留言记录*/
            2 usages
83      public Leaveword getLeaveword(int leaveWordId) throws Exception  {
84          Leaveword leaveword = leavewordMapper.getLeaveword(leaveWordId);
85          return leaveword;
86      }
87
88      /*更新留言记录*/
            1 usage
89      public void updateLeaveword(Leaveword leaveword) throws Exception {
90          leavewordMapper.updateLeaveword(leaveword);
91      }
92
93      /*删除一条留言记录*/
            1 usage
94      public void deleteLeaveword (int leaveWordId) throws Exception {
95          leavewordMapper.deleteLeaveword(leaveWordId);
96      }
97
98      /*删除多条留言信息*/
            1 usage
99  @   public int deleteLeavewords (String leaveWordIds) throws Exception {
100         String _leaveWordIds[] = leaveWordIds.split( regex: ",");
101         for(String _leaveWordId: _leaveWordIds) {
102             leavewordMapper.deleteLeaveword(Integer.parseInt(_leaveWordId));
103         }
104         return _leaveWordIds.length;
105     }
106
```

NewsService：

NewsService.java

```java
package com.chengxusheji.service;

import java.util.ArrayList;
import javax.annotation.Resource;
import org.springframework.stereotype.Service;
import com.chengxusheji.po.News;

import com.chengxusheji.mapper.NewsMapper;
2 usages
@Service
public class NewsService {

    9 usages
    @Resource NewsMapper newsMapper;
    /*每页显示记录数目*/
    6 usages
    private int rows = 10;;
    public int getRows() { return rows; }
    public void setRows(int rows) { this.rows = rows; }

    /*保存查询后总的页数*/
    4 usages
    private int totalPage;
    public void setTotalPage(int totalPage) { this.totalPage = totalPage; }
    public int getTotalPage() { return totalPage; }

    /*保存查询到的总记录数*/
    5 usages
    private int recordNumber;
    public void setRecordNumber(int recordNumber) { this.recordNumber = recordNumber; }
    public int getRecordNumber() { return recordNumber; }

    /*添加新闻信息记录*/
    1 usage
    public void addNews(News news) throws Exception {
        newsMapper.addNews(news);
    }

    /*按照查询条件分页查询新闻信息记录*/
    2 usages
    public ArrayList<News> queryNews(String newsTitle,String newsDate,int currentPage) throws Exception {
        String where = "where 1=1";
        if(!newsTitle.equals("")) where = where + " and t_news.newsTitle like '%" + newsTitle + "%'";
        if(!newsDate.equals("")) where = where + " and t_news.newsDate like '%" + newsDate + "%'";
        int startIndex = (currentPage-1) * this.rows;
        return newsMapper.queryNews(where, startIndex, this.rows);
    }

    /*按照查询条件查询所有记录*/
    1 usage
    public ArrayList<News> queryNews(String newsTitle,String newsDate) throws Exception  {
        String where = "where 1=1";
        if(!newsTitle.equals("")) where = where + " and t_news.newsTitle like '%" + newsTitle + "%'";
        if(!newsDate.equals("")) where = where + " and t_news.newsDate like '%" + newsDate + "%'";
        return newsMapper.queryNewsList(where);
    }

    /*查询所有新闻信息记录*/
    1 usage
    public ArrayList<News> queryAllNews()  throws Exception {
        return newsMapper.queryNewsList( where: "where 1=1");
    }

    /*当前查询条件下计算总的页数和记录数*/
    public void queryTotalPageAndRecordNumber(String newsTitle,String newsDate) throws Exception {
        String where = "where 1=1";
        if(!newsTitle.equals("")) where = where + " and t_news.newsTitle like '%" + newsTitle + "%'";
        if(!newsDate.equals("")) where = where + " and t_news.newsDate like '%" + newsDate + "%'";
        recordNumber = newsMapper.queryNewsCount(where);
        int mod = recordNumber % this.rows;
        totalPage = recordNumber / this.rows;
        if(mod != 0) totalPage++;
    }
```

```
77
78      /*根据主键获取新闻信息记录*/
        2 usages
79      public News getNews(int newsId) throws Exception  {
80          News news = newsMapper.getNews(newsId);
81          return news;
82      }
83
84      /*更新新闻信息记录*/
        1 usage
85      public void updateNews(News news) throws Exception {
86          newsMapper.updateNews(news);
87      }
88
89      /*删除一条新闻信息记录*/
        1 usage
90      public void deleteNews (int newsId) throws Exception {
91          newsMapper.deleteNews(newsId);
92      }
93
94      /*删除多条新闻信息信息*/
        1 usage
95  @   public int deleteNewss (String newsIds) throws Exception {
96          String _newsIds[] = newsIds.split( regex: ",");
97          for(String _newsId: _newsIds) {
98              newsMapper.deleteNews(Integer.parseInt(_newsId));
99          }
100         return _newsIds.length;
101     }
102 }
103
```

OrderInfoService：

OrderInfoService.java

```
1   package com.chengxusheji.service;
2
3   import java.util.ArrayList;
4   import javax.annotation.Resource;
5   import org.springframework.stereotype.Service;
6   import com.chengxusheji.po.UserInfo;
7   import com.chengxusheji.po.Doctor;
8   import com.chengxusheji.po.OrderInfo;
9
10  import com.chengxusheji.mapper.OrderInfoMapper;
    2 usages
11  @Service
12  public class OrderInfoService {
13
        9 usages
14      @Resource OrderInfoMapper orderInfoMapper;
15      /*每页显示记录数目*/
        6 usages
16      private int rows = 10;;
17      public int getRows() { return rows; }
20      public void setRows(int rows) { this.rows = rows; }
23
24      /*保存查询后总的页数*/
        4 usages
25      private int totalPage;
26      public void setTotalPage(int totalPage) { this.totalPage = totalPage; }
29      public int getTotalPage() { return totalPage; }
32
33      /*保存查询到的总记录数*/
        5 usages
34      private int recordNumber;
35      public void setRecordNumber(int recordNumber) { this.recordNumber = recordNumber; }
38      public int getRecordNumber() { return recordNumber; }
```

```
    /*添加预约信息记录*/
    1 usage
    public void addOrderInfo(OrderInfo orderInfo) throws Exception {
        orderInfoMapper.addOrderInfo(orderInfo);
    }

    /*按照查询条件分页查询预约信息记录*/
    2 usages
    public ArrayList<OrderInfo> queryOrderInfo(UserInfo userObj,Doctor doctorObj,String orderDate,String timeInterval,String telephone,String check
        String where = "where 1=1";
        if(null != userObj &&  userObj.getUser_name() != null  && !userObj.getUser_name().equals(""))  where += " and t_orderInfo.userObj='" + use
        if(null != doctorObj &&  doctorObj.getDoctorNumber() != null  && !doctorObj.getDoctorNumber().equals(""))  where += " and t_orderInfo.doct
        if(!orderDate.equals("")) where = where + " and t_orderInfo.orderDate like '%" + orderDate + "%'";
        if(!timeInterval.equals("")) where = where + " and t_orderInfo.timeInterval like '%" + timeInterval + "%'";
        if(!telephone.equals("")) where = where + " and t_orderInfo.telephone like '%" + telephone + "%'";
        if(!checkState.equals("")) where = where + " and t_orderInfo.checkState like '%" + checkState + "%'";
        if(!orderTime.equals("")) where = where + " and t_orderInfo.orderTime like '%" + orderTime + "%'";
        int startIndex = (currentPage-1) * this.rows;
        return orderInfoMapper.queryOrderInfo(where, startIndex, this.rows);
    }

    /*按照查询条件查询所有记录*/
    1 usage
    public ArrayList<OrderInfo> queryOrderInfo(UserInfo userObj,Doctor doctorObj,String orderDate,String timeInterval,String telephone,String che
        String where = "where 1=1";
        if(null != userObj &&  userObj.getUser_name() != null && !userObj.getUser_name().equals(""))  where += " and t_orderInfo.userObj='" + user
        if(null != doctorObj &&  doctorObj.getDoctorNumber() != null && !doctorObj.getDoctorNumber().equals(""))  where += " and t_orderInfo.docto
        if(!orderDate.equals("")) where = where + " and t_orderInfo.orderDate like '%" + orderDate + "%'";
        if(!timeInterval.equals("")) where = where + " and t_orderInfo.timeInterval like '%" + timeInterval + "%'";
        if(!telephone.equals("")) where = where + " and t_orderInfo.telephone like '%" + telephone + "%'";
        if(!checkState.equals("")) where = where + " and t_orderInfo.checkState like '%" + checkState + "%'";
        if(!orderTime.equals("")) where = where + " and t_orderInfo.orderTime like '%" + orderTime + "%'";
        return orderInfoMapper.queryOrderInfoList(where);
    }
```

```
    /*查询所有预约信息记录*/
    1 usage
    public ArrayList<OrderInfo> queryAllOrderInfo()  throws Exception {
        return orderInfoMapper.queryOrderInfoList( where: "where 1=1");
    }

    /*当前查询条件下计算总的页数和记录数*/
    public void queryTotalPageAndRecordNumber(UserInfo userObj,Doctor doctorObj,String orderDate,String timeInterval,String telephone,String checkS
        String where = "where 1=1";
        if(null != userObj &&  userObj.getUser_name() != null && !userObj.getUser_name().equals(""))  where += " and t_orderInfo.userObj='" + use
        if(null != doctorObj &&  doctorObj.getDoctorNumber() != null && !doctorObj.getDoctorNumber().equals(""))  where += " and t_orderInfo.doct
        if(!orderDate.equals("")) where = where + " and t_orderInfo.orderDate like '%" + orderDate + "%'";
        if(!timeInterval.equals("")) where = where + " and t_orderInfo.timeInterval like '%" + timeInterval + "%'";
        if(!telephone.equals("")) where = where + " and t_orderInfo.telephone like '%" + telephone + "%'";
        if(!checkState.equals("")) where = where + " and t_orderInfo.checkState like '%" + checkState + "%'";
        if(!orderTime.equals("")) where = where + " and t_orderInfo.orderTime like '%" + orderTime + "%'";
        recordNumber = orderInfoMapper.queryOrderInfoCount(where);
        int mod = recordNumber % this.rows;
        totalPage = recordNumber / this.rows;
        if(mod != 0) totalPage++;
    }

    /*根据主键获取预约信息记录*/
    2 usages
    public OrderInfo getOrderInfo(int orderId) throws Exception  {
        OrderInfo orderInfo = orderInfoMapper.getOrderInfo(orderId);
        return orderInfo;
    }
```

```
    /*更新预约信息记录*/
    1 usage
    public void updateOrderInfo(OrderInfo orderInfo) throws Exception {
        orderInfoMapper.updateOrderInfo(orderInfo);
    }

    /*删除一条预约信息记录*/
    1 usage
    public void deleteOrderInfo (int orderId) throws Exception {
        orderInfoMapper.deleteOrderInfo(orderId);
    }

    /*删除多条预约信息信息*/
    1 usage
    public int deleteOrderInfos (String orderIds) throws Exception {
        String _orderIds[] = orderIds.split( regex: ",");
        for(String _orderId: _orderIds) {
            orderInfoMapper.deleteOrderInfo(Integer.parseInt(_orderId));
        }
        return _orderIds.length;
    }
}
```

PatientService：

PatientService.java

```
package com.chengxusheji.service;

import java.util.ArrayList;
import javax.annotation.Resource;
import org.springframework.stereotype.Service;
import com.chengxusheji.po.Doctor;
import com.chengxusheji.po.Patient;

import com.chengxusheji.mapper.PatientMapper;
2 usages
@Service
public class PatientService {

    9 usages
    @Resource PatientMapper patientMapper;
    /*每页显示记录数目*/
    6 usages
    private int rows = 10;;
    public int getRows() { return rows; }
    public void setRows(int rows) { this.rows = rows; }

    /*保存查询后总的页数*/
    4 usages
    private int totalPage;
    public void setTotalPage(int totalPage) { this.totalPage = totalPage; }
    public int getTotalPage() { return totalPage; }

    /*保存查询到的总记录数*/
    5 usages
    private int recordNumber;
    public void setRecordNumber(int recordNumber) { this.recordNumber = recordNumber; }
    public int getRecordNumber() { return recordNumber; }

    /*添加病人信息记录*/
    1 usage
    public void addPatient(Patient patient) throws Exception {
        patientMapper.addPatient(patient);
    }
```

```java
    /*按照查询条件分页查询病人信息记录*/
    2 usages
    public ArrayList<Patient> queryPatient(Doctor doctorObj,String patientName,String cardNumber,String telephone,String addTime,int currentPage) t
        String where = "where 1=1";
        if(null != doctorObj &&  doctorObj.getDoctorNumber() != null  && !doctorObj.getDoctorNumber().equals(""))  where += " and t_patient.doctorO
        if(!patientName.equals("")) where = where + " and t_patient.patientName like '%" + patientName + "%'";
        if(!cardNumber.equals("")) where = where + " and t_patient.cardNumber like '%" + cardNumber + "%'";
        if(!telephone.equals("")) where = where + " and t_patient.telephone like '%" + telephone + "%'";
        if(!addTime.equals("")) where = where + " and t_patient.addTime like '%" + addTime + "%'";
        int startIndex = (currentPage-1) * this.rows;
        return patientMapper.queryPatient(where, startIndex, this.rows);
    }

    /*按照查询条件查询所有记录*/
    1 usage
    public ArrayList<Patient> queryPatient(Doctor doctorObj,String patientName,String cardNumber,String telephone,String addTime) throws Exceptio
        String where = "where 1=1";
        if(null != doctorObj &&  doctorObj.getDoctorNumber() != null && !doctorObj.getDoctorNumber().equals(""))  where += " and t_patient.doctorOb
        if(!patientName.equals("")) where = where + " and t_patient.patientName like '%" + patientName + "%'";
        if(!cardNumber.equals("")) where = where + " and t_patient.cardNumber like '%" + cardNumber + "%'";
        if(!telephone.equals("")) where = where + " and t_patient.telephone like '%" + telephone + "%'";
        if(!addTime.equals("")) where = where + " and t_patient.addTime like '%" + addTime + "%'";
        return patientMapper.queryPatientList(where);
    }

    /*查询所有病人信息记录*/
    1 usage
    public ArrayList<Patient> queryAllPatient()  throws Exception {
        return patientMapper.queryPatientList( where: "where 1=1");
    }

    /*当前查询条件下计算总的页数和记录数*/
    public void queryTotalPageAndRecordNumber(Doctor doctorObj,String patientName,String cardNumber,String telephone,String addTime) throws Excepti
        String where = "where 1=1";
        if(null != doctorObj &&  doctorObj.getDoctorNumber() != null && !doctorObj.getDoctorNumber().equals(""))  where += " and t_patient.doctorOb
        if(!patientName.equals("")) where = where + " and t_patient.patientName like '%" + patientName + "%'";
        if(!cardNumber.equals("")) where = where + " and t_patient.cardNumber like '%" + cardNumber + "%'";
        if(!telephone.equals("")) where = where + " and t_patient.telephone like '%" + telephone + "%'";
        if(!addTime.equals("")) where = where + " and t_patient.addTime like '%" + addTime + "%'";
        recordNumber = patientMapper.queryPatientCount(where);
        int mod = recordNumber % this.rows;
        totalPage = recordNumber / this.rows;
        if(mod != 0) totalPage++;
    }

    /*根据主键获取病人信息记录*/
    2 usages
    public Patient getPatient(int patientId) throws Exception  {
        Patient patient = patientMapper.getPatient(patientId);
        return patient;
    }

    /*更新病人信息记录*/
    1 usage
    public void updatePatient(Patient patient) throws Exception {
        patientMapper.updatePatient(patient);
    }

    /*删除一条病人信息记录*/
    1 usage
    public void deletePatient (int patientId) throws Exception {
        patientMapper.deletePatient(patientId);
    }

    /*删除多条病人信息信息*/
    1 usage
@   public int deletePatients (String patientIds) throws Exception {
        String _patientIds[] = patientIds.split( regex: ",");
        for(String _patientId: _patientIds) {
            patientMapper.deletePatient(Integer.parseInt(_patientId));
        }
        return _patientIds.length;
    }
}
```

UserInfoService：

UserInfoService.java

```java
package com.chengxusheji.service;

import java.util.ArrayList;
import javax.annotation.Resource;
import org.springframework.stereotype.Service;
import com.chengxusheji.po.UserInfo;

import com.chengxusheji.mapper.UserInfoMapper;
8 usages
@Service
public class UserInfoService {

    9 usages
    @Resource UserInfoMapper userInfoMapper;
    /*每页显示记录数目*/
    6 usages
    private int rows = 10;;
    public int getRows() { return rows; }
    public void setRows(int rows) { this.rows = rows; }

    /*保存查询后总的页数*/
    4 usages
    private int totalPage;
    public void setTotalPage(int totalPage) { this.totalPage = totalPage; }
    public int getTotalPage() { return totalPage; }

    /*保存查询到的总记录数*/
    5 usages
    private int recordNumber;
    public void setRecordNumber(int recordNumber) { this.recordNumber = recordNumber; }
    public int getRecordNumber() { return recordNumber; }

    /*添加用户记录*/
    1 usage
    public void addUserInfo(UserInfo userInfo) throws Exception {
        userInfoMapper.addUserInfo(userInfo);
    }
```

```java
    /*按照查询条件分页查询用户记录*/
    2 usages
    public ArrayList<UserInfo> queryUserInfo(String user_name,String name,String birthDate,String telephone,int currentPage) throws Exception {
        String where = "where 1=1";
        if(!user_name.equals("")) where = where + " and t_userInfo.user_name like '%" + user_name + "%'";
        if(!name.equals("")) where = where + " and t_userInfo.name like '%" + name + "%'";
        if(!birthDate.equals("")) where = where + " and t_userInfo.birthDate like '%" + birthDate + "%'";
        if(!telephone.equals("")) where = where + " and t_userInfo.telephone like '%" + telephone + "%'";
        int startIndex = (currentPage-1) * this.rows;
        return userInfoMapper.queryUserInfo(where, startIndex, this.rows);
    }

    /*按照查询条件查询所有记录*/
    1 usage
    public ArrayList<UserInfo> queryUserInfo(String user_name,String name,String birthDate,String telephone) throws Exception  {
        String where = "where 1=1";
        if(!user_name.equals("")) where = where + " and t_userInfo.user_name like '%" + user_name + "%'";
        if(!name.equals("")) where = where + " and t_userInfo.name like '%" + name + "%'";
        if(!birthDate.equals("")) where = where + " and t_userInfo.birthDate like '%" + birthDate + "%'";
        if(!telephone.equals("")) where = where + " and t_userInfo.telephone like '%" + telephone + "%'";
        return userInfoMapper.queryUserInfoList(where);
    }

    /*查询所有用户记录*/
    7 usages
    public ArrayList<UserInfo> queryAllUserInfo()  throws Exception {
        return userInfoMapper.queryUserInfoList( where: "where 1=1");
    }

    /*当前查询条件下计算总的页数和记录数*/
    public void queryTotalPageAndRecordNumber(String user_name,String name,String birthDate,String telephone) throws Exception {
        String where = "where 1=1";
        if(!user_name.equals("")) where = where + " and t_userInfo.user_name like '%" + user_name + "%'";
        if(!name.equals("")) where = where + " and t_userInfo.name like '%" + name + "%'";
        if(!birthDate.equals("")) where = where + " and t_userInfo.birthDate like '%" + birthDate + "%'";
        if(!telephone.equals("")) where = where + " and t_userInfo.telephone like '%" + telephone + "%'";
```

```
        recordNumber = userInfoMapper.queryUserInfoCount(where);
        int mod = recordNumber % this.rows;
        totalPage = recordNumber / this.rows;
        if(mod != 0) totalPage++;
    }

    /*根据主键获取用户记录*/
    3 usages
    public UserInfo getUserInfo(String user_name) throws Exception  {
        UserInfo userInfo = userInfoMapper.getUserInfo(user_name);
        return userInfo;
    }

    /*更新用户记录*/
    1 usage
    public void updateUserInfo(UserInfo userInfo) throws Exception {
        userInfoMapper.updateUserInfo(userInfo);
    }

    /*删除一条用户记录*/
    1 usage
    public void deleteUserInfo (String user_name) throws Exception {
        userInfoMapper.deleteUserInfo(user_name);
    }

    /*删除多条用户信息*/
    1 usage
    public int deleteUserInfos (String user_names) throws Exception {
        String _user_names[] = user_names.split( regex: ",");
        for(String _user_name: _user_names) {
            userInfoMapper.deleteUserInfo(_user_name);
        }
        return _user_names.length;
    }
}
```

util 包：

ExportExcelUtil：

ExportExcelUtil.java

```java
package com.chengxusheji.utils;

import java.io.BufferedInputStream;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.util.Collection;
import java.util.Iterator;

import org.apache.poi.hssf.usermodel.HSSFCell;
import org.apache.poi.hssf.usermodel.HSSFCellStyle;
import org.apache.poi.hssf.usermodel.HSSFClientAnchor;
import org.apache.poi.hssf.usermodel.HSSFComment;
import org.apache.poi.hssf.usermodel.HSSFFont;
import org.apache.poi.hssf.usermodel.HSSFPatriarch;
import org.apache.poi.hssf.usermodel.HSSFRichTextString;
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.hssf.util.HSSFColor;

21 usages
public class ExportExcelUtil {

    public void exportExcel(Collection<String[]> dataset, OutputStream out) {
        exportExcel( rootPath: "", title: "测试POI导出EXCEL文档",  headers: null, dataset, out,  pattern: "yyyy-MM-dd");
    }

    public void exportExcel(String rootPath,String title, String[] headers,
            Collection<String[]> dataset, OutputStream out) {
        exportExcel(rootPath,title, headers, dataset, out,  pattern: "yyyy-MM-dd");
    }

    public void exportExcel(String[] headers, Collection<String[]> dataset,
            OutputStream out, String pattern) {
        exportExcel( rootPath: "", title: "测试POI导出EXCEL文档", headers, dataset, out, pattern);
    }

    /**
     * 这是一个通用的方法，可以将放置在JAVA集合中并且符号一定条件的数据以EXCEL 的形式输出到指定IO设备上
     *
     * @param title
     *            表格标题名
     * @param headers
     *            表格属性列名数组
     * @param dataset
     *            需要显示的数据集合
     * @param out
     *            与输出设备关联的流对象，可以将EXCEL文档导出到本地文件或者网络中
     * @param pattern
     *            如果有时间数据，设定输出格式。默认为"yyy-MM-dd"
     */
    /unchecked/
@   public void exportExcel(String rootPath,String title, String[] headers,
            Collection<String[]> dataset, OutputStream out, String pattern) {
        // 声明一个工作薄
        HSSFWorkbook workbook = new HSSFWorkbook();
        // 生成一个表格
        HSSFSheet sheet = workbook.createSheet(title);
        // 设置表格默认列宽度为15个字节
        sheet.setDefaultColumnWidth((short) 15);
        // 生成一个样式
        HSSFCellStyle style = workbook.createCellStyle();
        // 设置这些样式
        style.setFillForegroundColor(HSSFColor.SKY_BLUE.index);
        style.setFillPattern(HSSFCellStyle.SOLID_FOREGROUND);
        style.setBorderBottom(HSSFCellStyle.BORDER_THIN);
        style.setBorderLeft(HSSFCellStyle.BORDER_THIN);
        style.setBorderRight(HSSFCellStyle.BORDER_THIN);
        style.setBorderTop(HSSFCellStyle.BORDER_THIN);
        style.setAlignment(HSSFCellStyle.ALIGN_CENTER);
        // 生成一个字体
        HSSFFont font = workbook.createFont();
```

```
        font.setColor(HSSFColor.VIOLET.index);
        font.setFontHeightInPoints((short) 12);
        font.setBoldweight(HSSFFont.BOLDWEIGHT_BOLD);
        // 把字体应用到当前的样式
        style.setFont(font);
        // 生成并设置另一个样式
        HSSFCellStyle style2 = workbook.createCellStyle();
        style2.setFillForegroundColor(HSSFColor.LIGHT_YELLOW.index);
        style2.setFillPattern(HSSFCellStyle.SOLID_FOREGROUND);
        style2.setBorderBottom(HSSFCellStyle.BORDER_THIN);
        style2.setBorderLeft(HSSFCellStyle.BORDER_THIN);
        style2.setBorderRight(HSSFCellStyle.BORDER_THIN);
        style2.setBorderTop(HSSFCellStyle.BORDER_THIN);
        style2.setAlignment(HSSFCellStyle.ALIGN_CENTER);
        style2.setVerticalAlignment(HSSFCellStyle.VERTICAL_CENTER);
        // 生成另一个字体
        HSSFFont font2 = workbook.createFont();
        font2.setBoldweight(HSSFFont.BOLDWEIGHT_NORMAL);
        // 把字体应用到当前的样式
        style2.setFont(font2);

        // 声明一个画图的顶级管理器
        HSSFPatriarch patriarch = sheet.createDrawingPatriarch();
        // 定义注释的大小和位置,详见文档
        HSSFComment comment = patriarch.createComment(new HSSFClientAnchor( dx1: 0,
                dy1: 0,  dx2: 0,  dy2: 0, (short) 4,  row1: 2, (short) 6,  row2: 5));
        // 设置注释内容
        comment.setString(new HSSFRichTextString("可以在POI中添加注释！"));
        // 设置注释作者，当鼠标移动到单元格上是可以在状态栏中看到该内容.
        comment.setAuthor("leno");

        // 产生表格标题行
        HSSFRow row = sheet.createRow( rownum: 0);
        for (short i = 0; i < headers.length; i++) {
            HSSFCell cell = row.createCell(i);
            cell.setCellStyle(style);
            HSSFRichTextString text = new HSSFRichTextString(headers[i]);
            cell.setCellValue(text);
        }

        // 遍历集合数据，产生数据行
        Iterator<String[]> it = dataset.iterator();
        int index = 0;
        while (it.hasNext()) {
            index++;
            row = sheet.createRow(index);
            String[] t = (String[]) it.next();
            for (short i = 0; i < t.length; i++) {
                HSSFCell cell = row.createCell(i);
                cell.setCellStyle(style2);
                // 判断值的类型后进行强制类型转换
                String textValue = t[i];

                if(textValue.startsWith("upload/"))
                {
                    // 有图片时，设置行高为50px;
                    row.setHeightInPoints(50);
                    // 设置图片所在列宽度为80px,注意这里单位的一个换算
                    sheet.setColumnWidth(i, (short) (35.7 * 80)); //
                    //sheet.autoSizeColumn(i);
                    BufferedInputStream bis;
                    byte[] buf = null;
                    try {
                        bis = new BufferedInputStream(
                                new FileInputStream( name: rootPath + textValue));
                        buf = new byte[bis.available()];
                        while ((bis.read(buf)) != -1) {}

                        HSSFClientAnchor anchor = new HSSFClientAnchor( dx1: 0,  dy1: 0,  dx2: 1023,
                                dy2: 255, (short) i, index, (short) i, index);
                        anchor.setAnchorType(2);
                        patriarch.createPicture(anchor,
                                workbook.addPicture( buf,
```

```
                    HSSFWorkbook.PICTURE_TYPE_JPEG));
                } catch (Exception e) {
                    e.printStackTrace();
                }

            } else {
                HSSFRichTextString richString = new HSSFRichTextString(
                        textValue);
                HSSFFont font3 = workbook.createFont();
                font3.setColor(HSSFColor.BLUE.index);
                richString.applyFont(font3);
                cell.setCellValue(richString);
            }

            /*
             * else if (value instanceof byte[]) { // 有图片时，设置行高为60px;
             * row.setHeightInPoints(60); // 设置图片所在列宽度为80px,注意这里单位的一个换算
             * sheet.setColumnWidth(i, (short) (35.7 * 80)); //
             * sheet.autoSizeColumn(i); byte[] bsValue = (byte[]) value;
             * HSSFClientAnchor anchor = new HSSFClientAnchor(0, 0, 1023,
             * 255, (short) 6, index, (short) 6, index);
             * anchor.setAnchorType(2); patriarch.createPicture(anchor,
             * workbook.addPicture( bsValue,
             * HSSFWorkbook.PICTURE_TYPE_JPEG)); } else{ //其它数据类型都当作字符串简单处理
             * textValue = value.toString(); }
             */

        }
    }
```

```
    try {
        workbook.write(out);
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

}
}
```

UserException：

UserException.java

```java
package com.chengxusheji.utils;

public class UserException extends RuntimeException {

    /**
     *
     */
    no usages
    private static final long serialVersionUID = 1L;

    public UserException() {
        super();
        // TODO Auto-generated constructor stub
    }

    public UserException(String message, Throwable cause) {
        super(message, cause);
        // TODO Auto-generated constructor stub
    }

    public UserException(String message) {
        super(message);
        // TODO Auto-generated constructor stub
    }

    public UserException(Throwable cause) {
        super(cause);
        // TODO Auto-generated constructor stub
    }

```

Web.xml 配置文件:

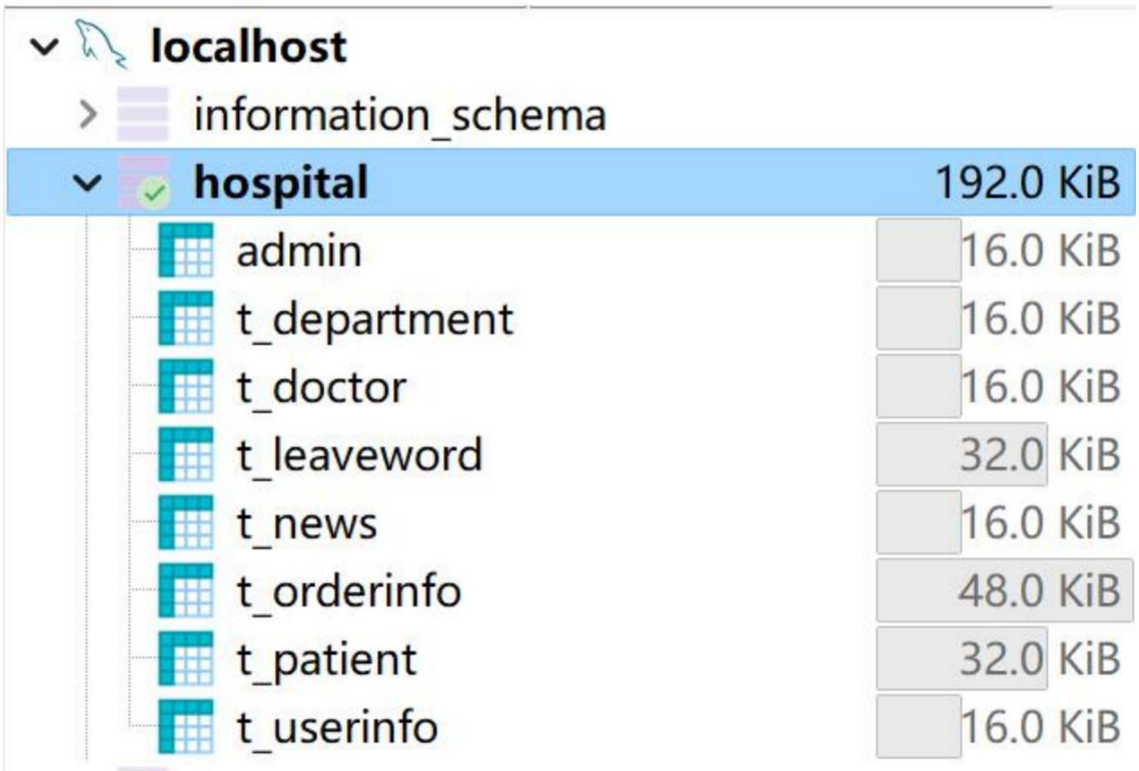
web.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">

    <!-- 指定spring的配置文件，默认从web根目录寻找配置文件，我们可以通过spring提供的classpath:前缀指定从类路径下寻找 -->
    <context-param>
       <param-name>contextConfigLocation</param-name>
       <param-value>/WEB-INF/classes/spring/applicationContext-*.xml</param-value>
    </context-param>


    <!-- 对Spring容器进行实例化 -->
    <listener>
         <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>


    <!-- springmvc前端控制器，rest配置 -->
    <servlet>
        <servlet-name>springmvc</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <!-- contextConfigLocation配置springmvc加载的配置文件（配置处理器映射器、适配器等等） 如果不配置contextConfigLocation，默认加
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>classpath:spring/springmvc.xml</param-value>
        </init-param>
    </servlet>
    <servlet-mapping>
        <servlet-name>springmvc</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
```

```
    <filter>
        <filter-name>CharacterFilter</filter-name>
        <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
        <init-param>
            <param-name>encoding</param-name>
            <param-value>UTF-8</param-value>
        </init-param>
    </filter>
    <filter-mapping>
        <filter-name>CharacterFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>


  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
</web-app>

```

3. JavaWeb 项目运行

首先运行前数据库各个表的情况如下:



admin 表:

id	username	password	type
1	a	a	0
2	b	b	(NULL)
3	c	c	(NULL)
4	d	d	1
5	e	e	2
6	k	k	1
7	h	h	(NULL)

department 表：

hospital.t_department: 4 总记录数 (大约)

departmentId	departmentName	departmentDesc	birthDate	chargeMan
2	内科	内科内科内科	2023-12-30	内科
3	外科	外科	2023-12-30	外科
4	皮肤科	皮肤科皮肤科	2023-12-30	皮肤科
5	dddd	dfsdfsfds	2023-12-30	aaaaa

doctor 表：

id	departmentObj	doctorName	sex	doctorPhoto	birthDate	position	experience	contactWay	goodAt	doctorDesc	adminid	doctorNumber	password
8	2	d	女	upload/NoImage.jpg	2023-12-30	d	d	d	d	dd	4	d	d

leaveword 表：

leaveWordId	leaveTitle	leaveContent	userObj	leaveTime	replyContent	replyTime
1	留言	留言留言留言	e	2023-12-30 21:20:04	回复回复	2023-12-30 21:21:32
(NULL)				(NULL)	(NULL)	(NULL)

news 表：

newsId	newsTitle	newsPhoto	newsContent	newsDate	newsFrom
1	新闻	upload/NoImage.jpg	新闻新闻新闻新闻	2023-12-30	百度

orderinfo 表：

orderId	userObj	doctorObj	orderDate	timeInterval	telephone	orderTime	checkState	replyContent
2	e	d	2023-12-30	12:00	13123	2023-12-30 21:19:35	通过	可以来

patient 表：

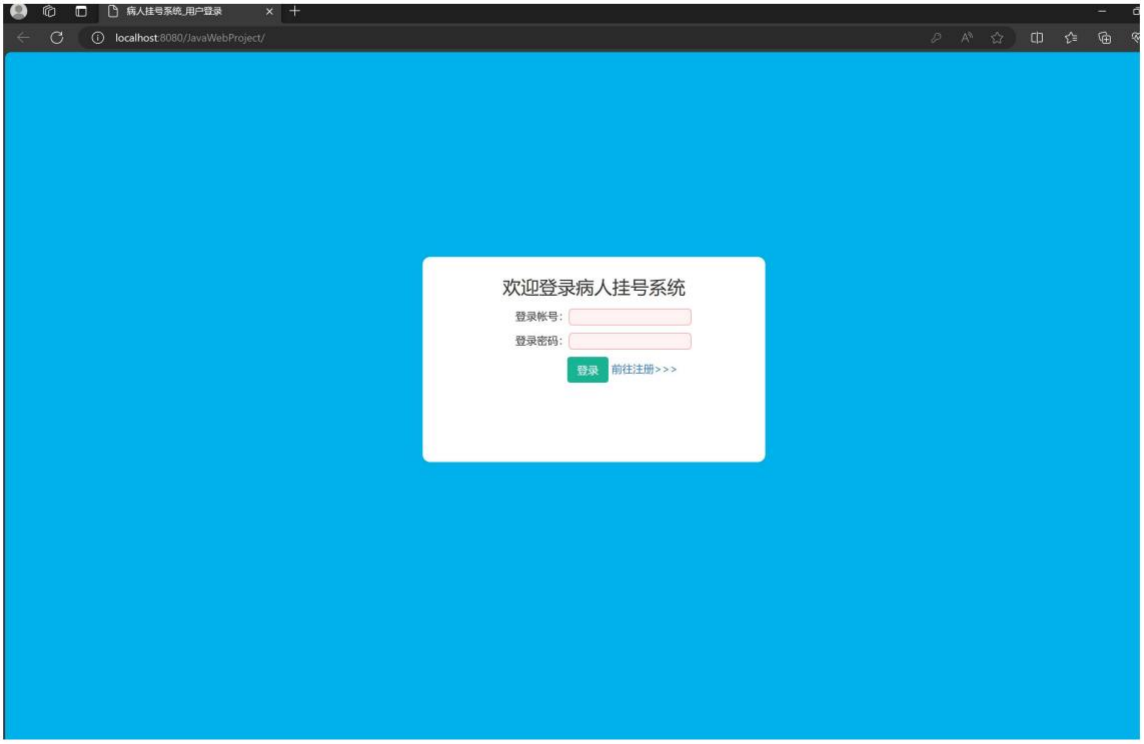
patientId	doctorObj	patientName	sex	cardNumber	telephone	illnessCase	addTime

userinfo 表：

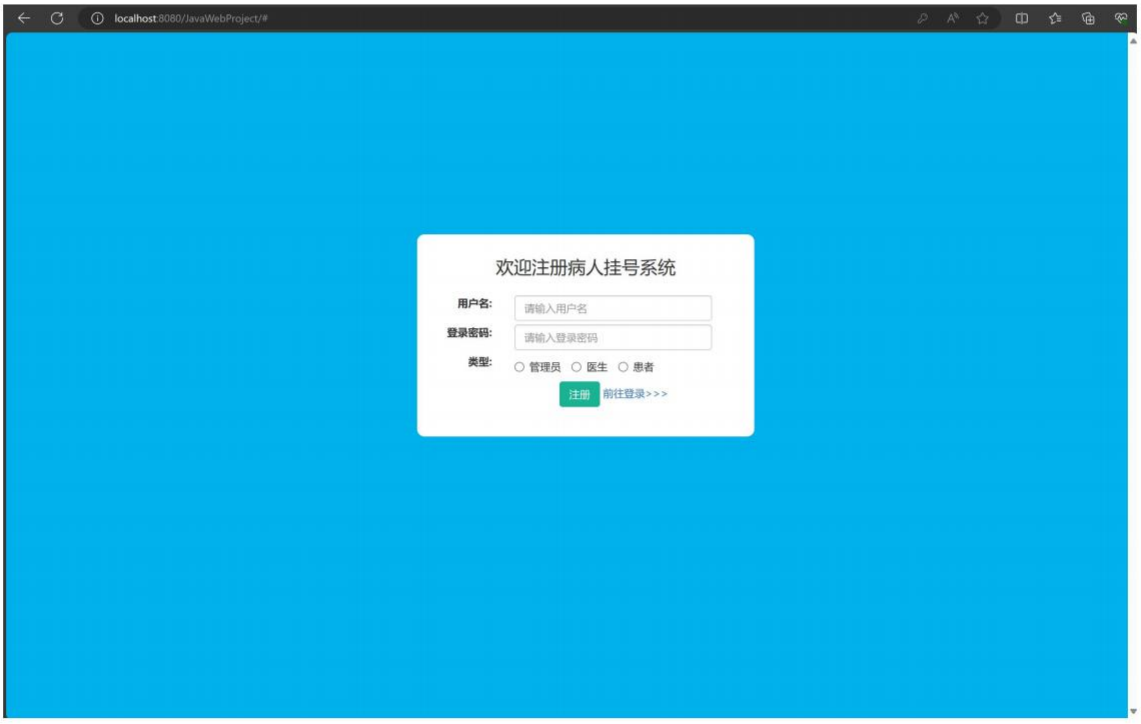
id	name	gender	birthDate	userPhoto	telephone	email	address	regTime	adminid	user_name	password
1	e	男	2023-12-30	upload/NoImage.jpg	131	1313	123	2023-12-30 21:18:41	5	e	e

正常运行各功能展示：

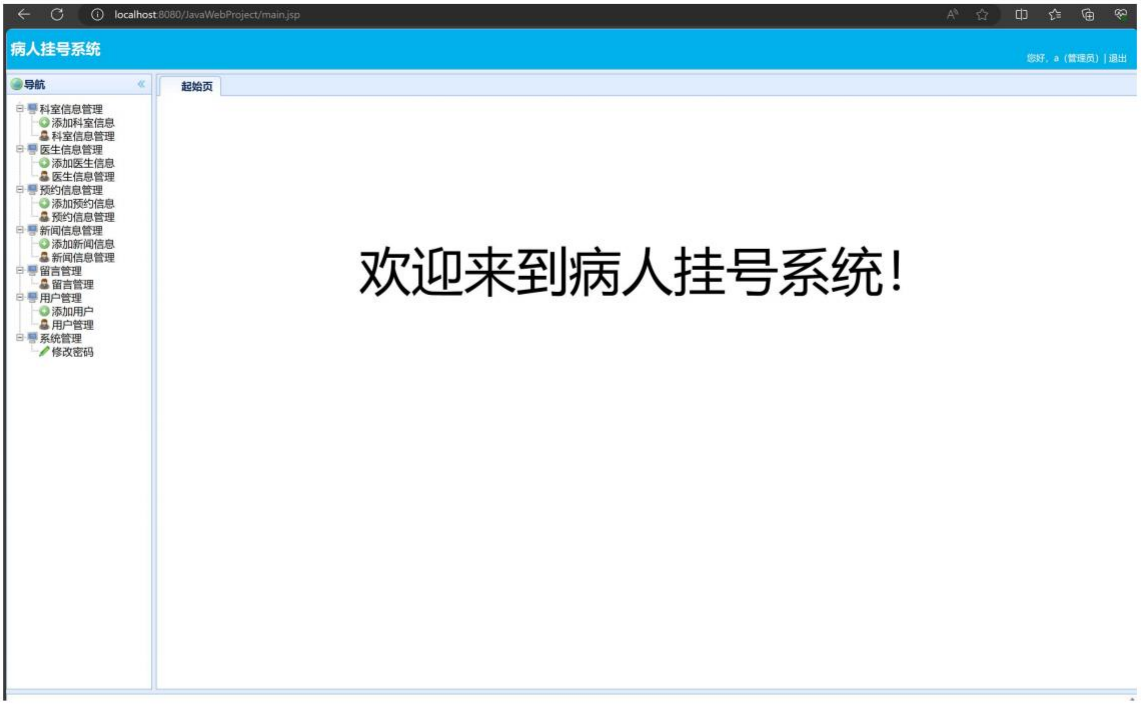
首先进入登陆页面，这时可以直接登陆或者进行注册



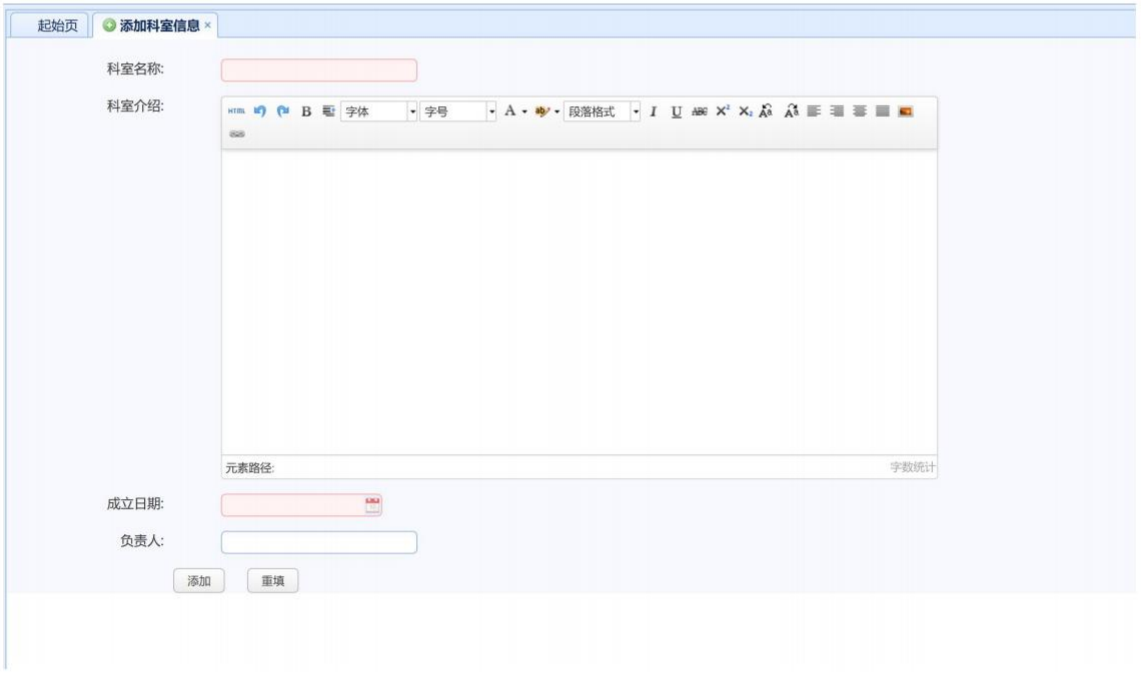
点击不同的单选按钮会出现不同的注册信息填写要求（管理员、医生、患者），这里选择管理员身份进行后续流程：



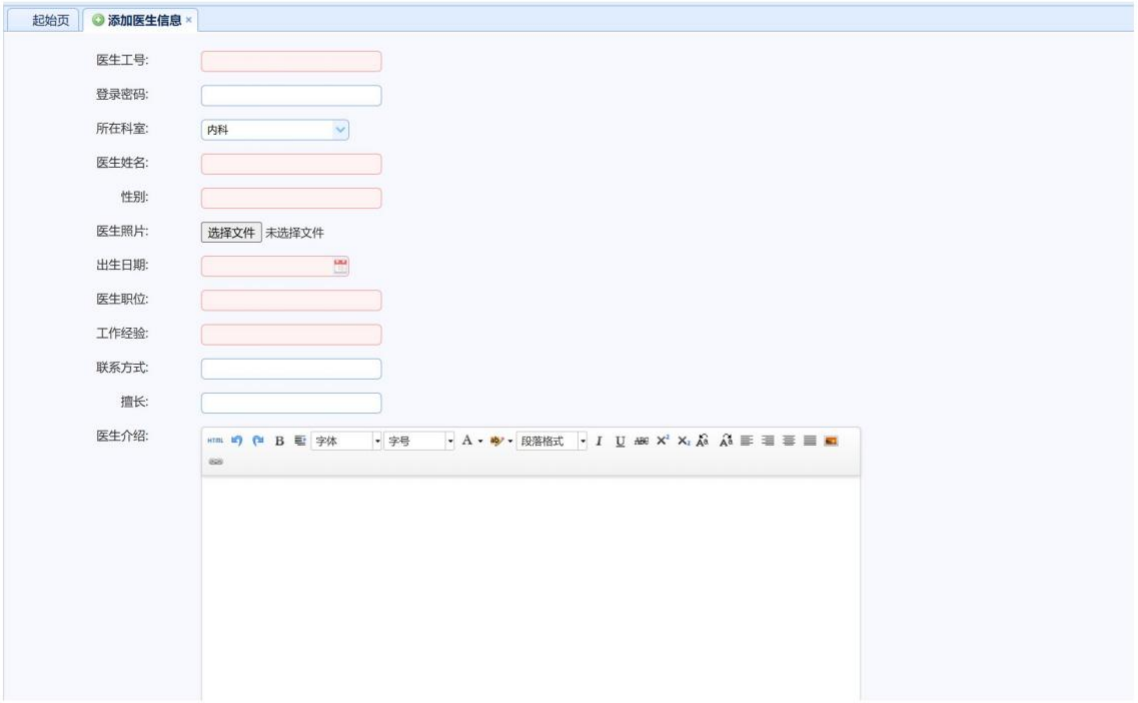
管理员登录后跳转至界面：



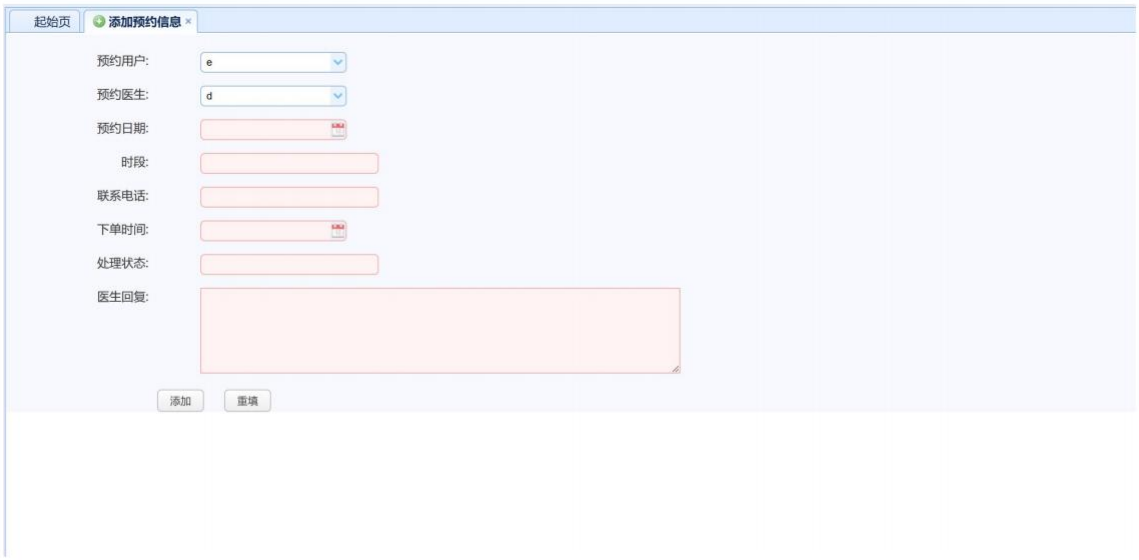
登录后可管理科室信息、管理病人信息、管理预约信息、管理新闻信息、管理留言、管理用户、管理系统（修改密码）




起始页
添加科室信息
科室信息管理
修改
删除
刷新
取消选择
导出到excel
科室名称:
成立日期:
查询
科室id
科室名称
成立日期
负责人
1 2 内科 2023-12-30 内科
2 3 外科 2023-12-30 外科
3 4 皮肤科 2023-12-30 皮肤科
4 5 dddd 2023-12-30 aaaaa

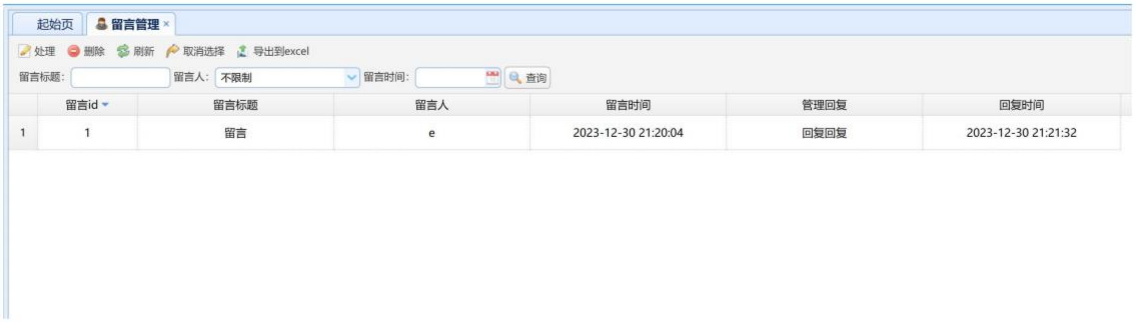

起始页
添加医生信息
医生工号:
登录密码:
所在科室:
内科
医生姓名:
性别:
医生照片:
选择文件
未选择文件
出生日期:
医生职位:
工作经验:
联系方式:
擅长:
医生介绍:
字体
字号
段落格式

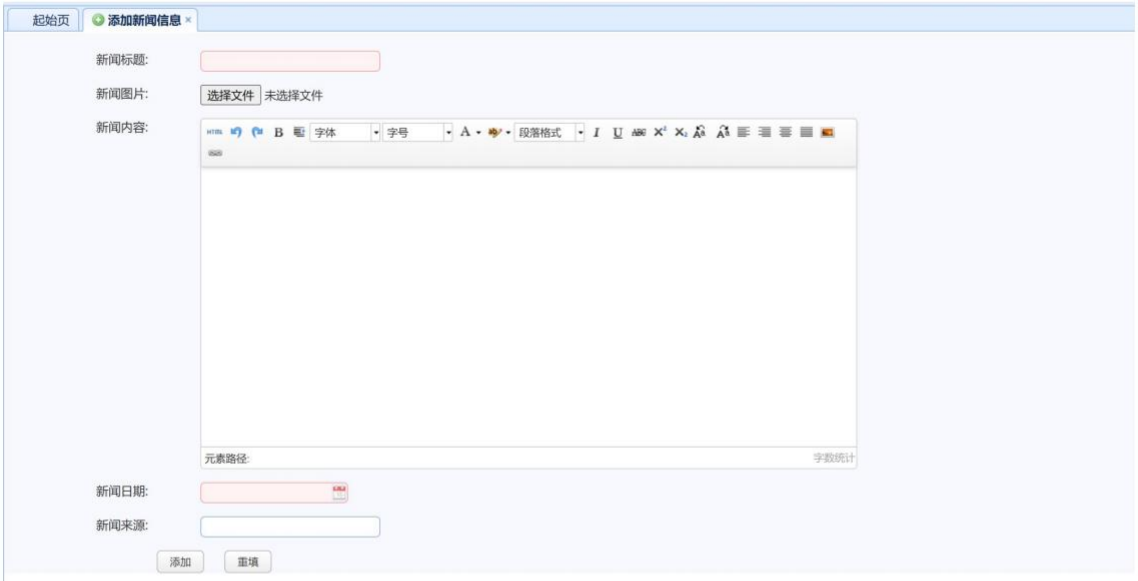

起始页
医生信息管理
修改
删除
刷新
取消选择
导出到excel
医生工号:
所在科室:
不限制
医生姓名:
出生日期:
查询
医生工号
所在科室
医生姓名
性别
医生照片
出生日期
医生职位
工作经验
联系方式
1 d 内科 d 女 2023-12-30 d d d


起始页
添加预约信息
预约用户:
e
预约医生:
d
预约日期:
时段:
联系电话:
下单时间:
处理状态:
医生回复:
添加
重填


起始页
添加预约信息
预约信息管理
处理
删除
刷新
取消选择
导出到excel
预约用户:
不限制
预约医生:
不限制
预约日期:
时段:
联系电话:
处理状态:
下单时间:
查询
预约id
预约用户
预约医生
预约日期
时段
联系电话
下单时间
处理状态
医生回复
1
2
e
d
2023-12-30
12:00
13123
2023-12-30 21:19:35
通过
可以来


起始页
添加用户
用户名:
登录密码:
姓名:
性别:
出生日期:
用户照片:
选择文件
未选择文件
联系电话:
邮箱:
家庭地址:
注册时间:
添加
重填


起始页
留言管理
处理
删除
刷新
取消选择
导出到excel
留言标题:
留言人:
不限制
留言时间:
查询
留言id
留言标题
留言人
留言时间
管理回复
回复时间
1
1
留言
e
2023-12-30 21:20:04
回复回复
2023-12-30 21:21:32


起始页
添加新闻信息
新闻标题:
新闻图片:
选择文件
未选择文件
新闻内容:
字体
字号
段落格式
元素路径:
字数统计
新闻日期:
新闻来源:
添加
重填


起始页
添加新闻信息
新闻信息管理
修改
删除
刷新
取消选择
导出到excel
新闻标题:
新闻日期:
查询
新闻id
新闻标题
新闻图片
新闻日期
新闻来源
1
1
新闻
暂无图片
2023-12-30
百度

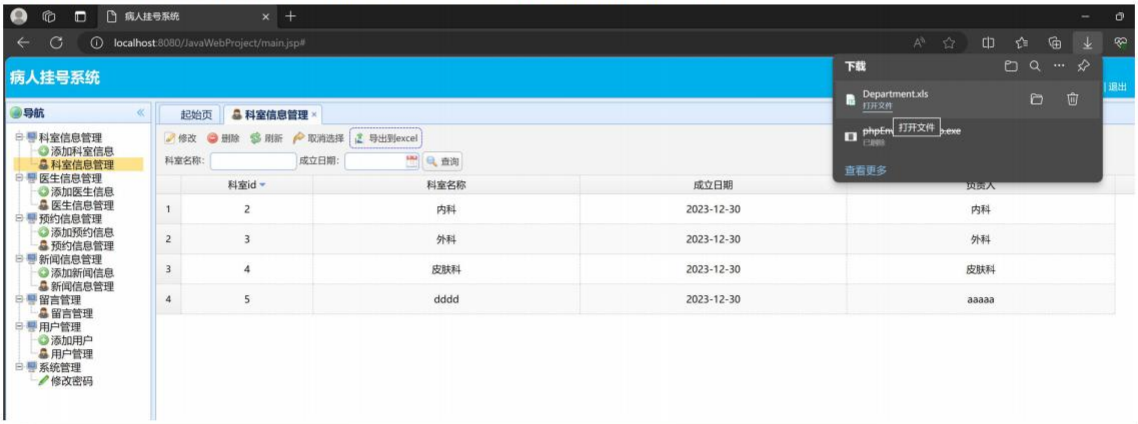

起始页
添加用户
用户管理
修改
删除
刷新
取消选择
导出到excel
用户名:
姓名:
出生日期:
联系电话:
查询
ID
用户名
姓名
性别
出生日期
用户照片
联系电话
邮箱
注册时间
1
1
e
e
男
2023-12-30
暂无图片
131
1313
2023-12-30 21:18:41

该系统其他页面以及细节展示：

附加实验部分：

附加实验总体来说在实验六的基础上有所增加，代码在上文已有所体现。

点击导出到 excel，会自动生成 excel 表并保存到指定地方：



四、心得体会

这次的实验耗费了大量的时间，我在这次实验中遇到的最大问题可能就是对一个 web 项目结构的不熟悉，我不是很清楚，什么类应该做什么事情，在之前也由于这个原因犯了很多错误，导致到最后要更改自己的一些类的结构，比较麻烦。不过好在实验课上师兄师姐和我们讲解了很多，帮助我们理清思路，然后让我慢慢在自己动手操作的过程中，对 web 项目结构逐渐清晰。

这次实验我比往常的实验花费了更多的时间，过程也更加艰辛。例如在运行之后输入了正确的账号密码，但一直无法登录进去。就这一问题我询问了我同为计算机专业的叔叔，他帮我远程调试代码，最后告诉我是因为我数据库驱动可能有问题，始终加载不上，于是他帮我安装了新的数据库，解决了这一问题。

虽然这次实验的结果存在着不少的瑕疵，也有很多需要改善的地方，因为成功完成实验的时间较晚，所以在实验报告的撰写方面，显得十分仓促，在很多地方也缺少文字说明。但我很清楚自己在这次的实验中有很大的成长与进步空间，这对我来说，远远比做出完美的实验结果和报告来得重要得多。理论和实践能力是相辅相成的，只有将理论知识应用到实际操作中，才能真正理解和掌握它们。从这次实验中我也体会到了 java 面向网络编程的优势，以及 java 网络编程的整个流程，我也开始对 java 网络编程有了很多的兴趣，希望以后可以更多地学习相关方面的知识。