

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

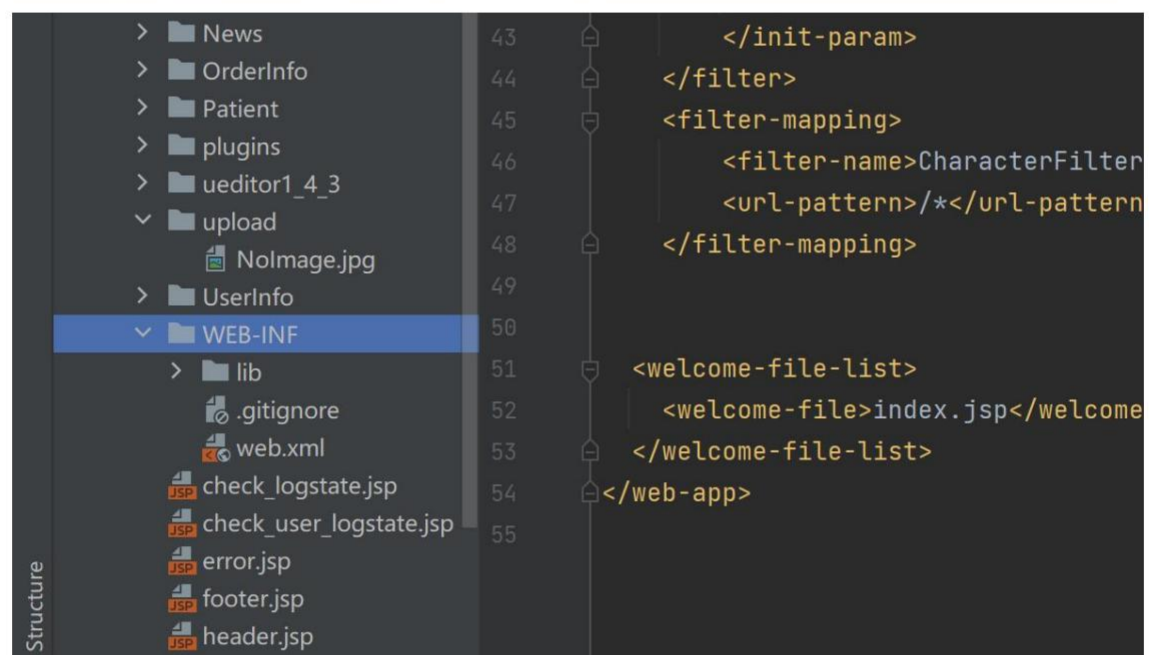
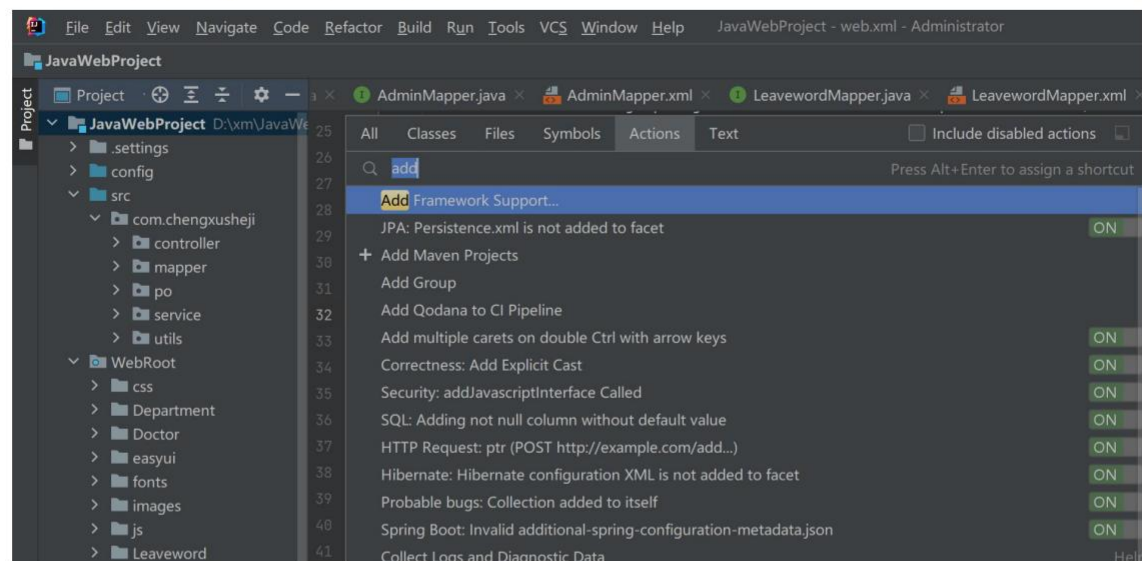
三、实验步骤及其结果截图展示

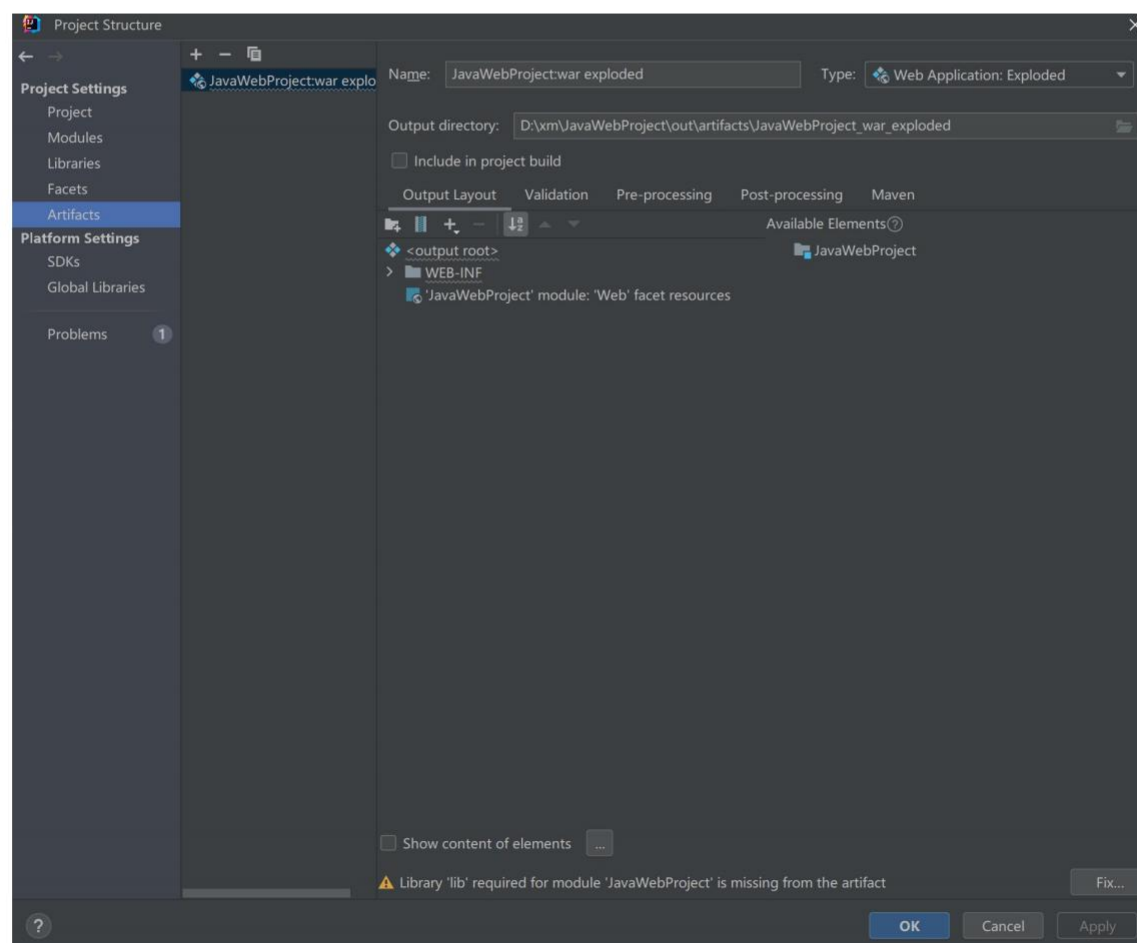
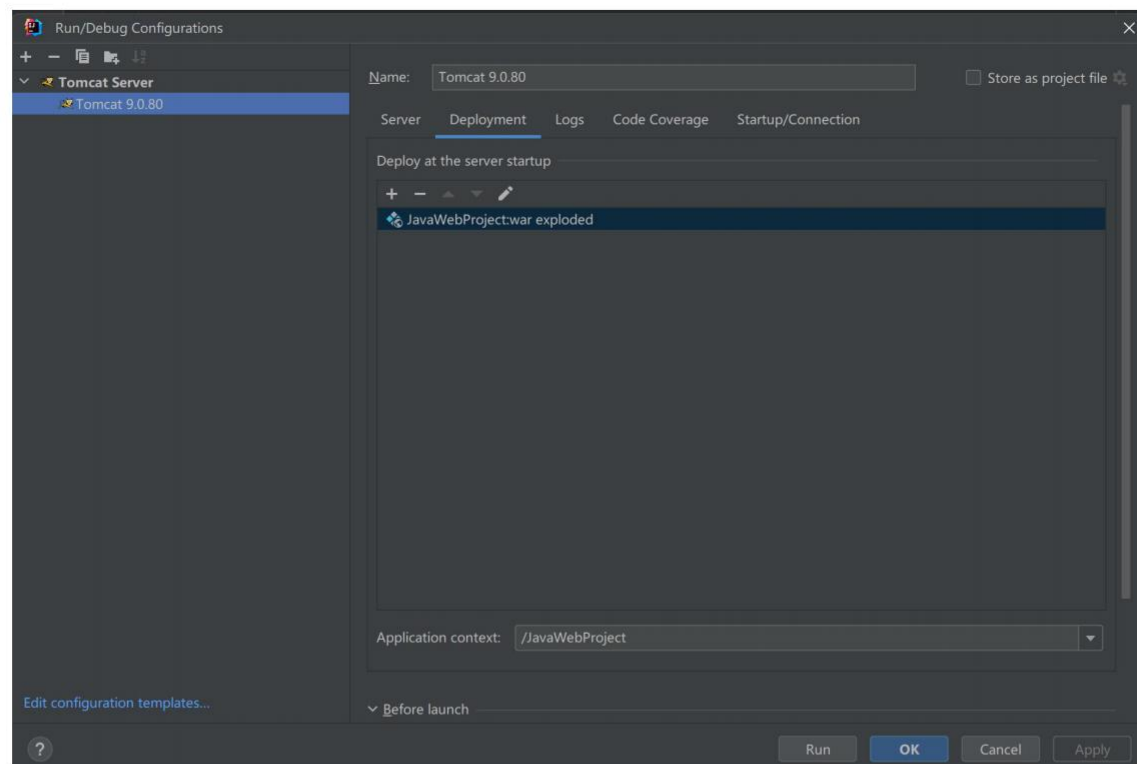
本次实验部分：

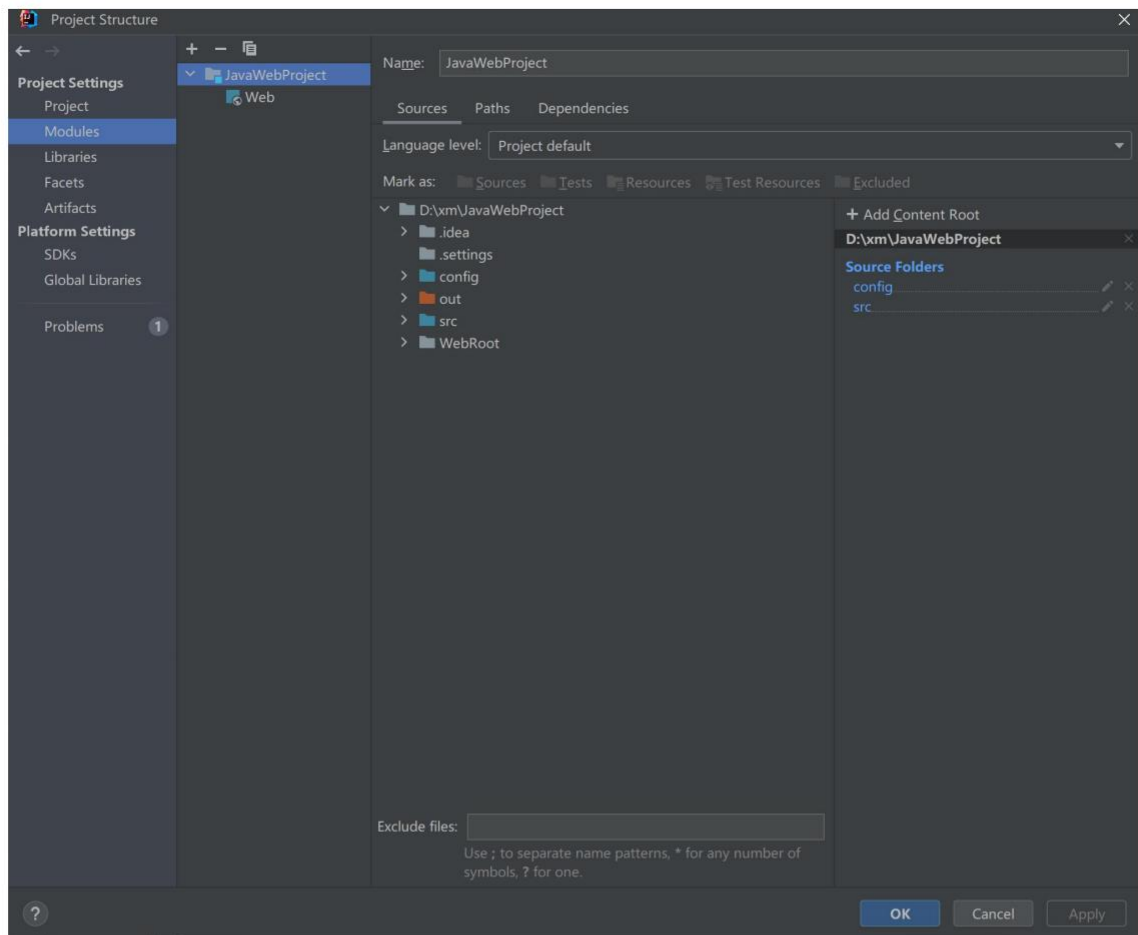
1.JavaWeb 项目的创建流程

依照流程操作如下：

创建普通 maven 项目，add framework 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。







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
1 package com.chengxusheji.controller;
2
3 import java.beans.PropertyEditorSupport;
4 import java.io.File;
5 import java.io.IOException;
6 import java.io.PrintWriter;
7 import java.text.SimpleDateFormat;
8 import java.util.Date;
9 import java.util.UUID;
10
11 import javax.servlet.http.HttpServletRequest;
12 import javax.servlet.http.HttpServletResponse;
13
14 import org.json.JSONException;
15 import org.json.JSONObject;
16 import org.springframework.beans.propertyeditors.CustomDateEditor;
17 import org.springframework.web.bind.WebDataBinder;
18 import org.springframework.web.bind.annotation.InitBinder;
19 import org.springframework.web.multipart.MultipartFile;
20 import org.springframework.web.multipart.MultipartHttpServletRequest;
21
22 import com.chengxusheji.utils.UserException;
23
24 8 usages 8 inheritors
25 public class BaseController {
26
27
28     @InitBinder
29     // 必须有一个参数WebDataBinder
30     public void initBinder(WebDataBinder binder) {
31         //System.out.println(binder.getFieldDefaultPrefix());
32         binder.registerCustomEditor(Date.class, new CustomDateEditor(
33             new SimpleDateFormat( pattern: "yyyy-MM-dd"), allowEmpty: false));
34
35         binder.registerCustomEditor(Integer.class, new PropertyEditorSupport() {
36             no usages
37             @Override
```

```

37  public String getAsText() { return (getValue() == null) ? "" : getValue().toString(); }
    no usages
40  @Override
41  public void setAsText(String text) {
42      Integer value = null;
43      if (null != text && !text.equals("")) {
44          try {
45              value = Integer.valueOf(text);
46          } catch (Exception ex) {
47              throw new UserException("数据格式输入不正确!");
48          }
49      }
50      setValue(value);
51  }
52  });
53
54  //binder.registerCustomEditor(Integer.class, null, new CustomNumberEditor(Integer.class, null, true));
55
56  binder.registerCustomEditor(Float.class, new PropertyEditorSupport() {
57      no usages
58      @Override
59      public String getAsText() { return (getValue() == null) ? "" : getValue().toString(); }
60      no usages
61      @Override
62      public void setAsText(String text) {
63          Float value = null;
64          if (null != text && !text.equals("")) {
65              try {
66                  value = Float.valueOf(text);
67              } catch (Exception e) {
68                  throw new UserException("数据格式输入不正确!");
69              }
70          }
71          setValue(value);
72      }
73  });
74  }

```

```

76  /**
77   * 处理图片文件上传，返回保存的文件名路径
78   * fileKeyName: 图片上传表单key
79   * @throws IOException
80   * @throws IllegalStateException
81   */
82  6 usages
83  @ public String handlePhotoUpload(HttpServletRequest request, String fileKeyName) throws IllegalStateException, IOException {
84      String fileName = "upload/NoImage.jpg";
85      MultipartHttpServletRequest multipartRequest = (MultipartHttpServletRequest) request;
86      /** 构建图片保存的目录**/
87      String photoBookPathDir = "/upload";
88      /** 得到图片保存目录的真实路径**/
89      String photoBookRealPathDir = request.getSession().getServletContext().getRealPath(photoBookPathDir);
90      /** 根据真实路径创建目录**/
91      File photoBookSaveFile = new File(photoBookRealPathDir);
92      if (!photoBookSaveFile.exists())
93          photoBookSaveFile.mkdirs();
94      /** 页面控件的文件流**/
95      MultipartFile multipartFile_photoBook = multipartRequest.getFile(fileKeyName);
96      if (!multipartFile_photoBook.isEmpty()) {
97          /** 获取文件的后缀**/
98          String suffix = multipartFile_photoBook.getOriginalFilename().substring
99              (multipartFile_photoBook.getOriginalFilename().lastIndexOf( str. "." ));
100          String smallSuffix = suffix.toLowerCase();
101          if (!smallSuffix.equals(".jpg") && !smallSuffix.equals(".gif") && !smallSuffix.equals(".png"))
102              throw new UserException("图片格式不正确!");
103          /** 使用UUID生成文件名称**/
104          String photoBookFileName = UUID.randomUUID().toString() + suffix; // 构建文件名称
105          //String logImageName = multipartFile.getOriginalFilename();
106          /** 构成完整的文件保存路径加文件**/
107          String photoBookFilePath = photoBookRealPathDir + File.separator + photoBookFileName;
108          File photoBookFile = new File(photoBookFilePath);
109          multipartFile_photoBook.transferTo(photoBookFile);
110      }

```

```

110
111     fileName = "upload/" + photoBookFileName;
112 }
113
114     return fileName;
115 }
116
117
118 /**
119  * 处理图片文件上传，返回保存的文件名路径
120  * fileKeyName: 图片上传表单key
121  * @throws IOException
122  * @throws IllegalStateException
123  */
124 no usages
125 @ public String handleFileUpload(HttpServletRequest request,String fileKeyName) throws IllegalStateException, IOException {
126     String fileName = "";
127     MultipartHttpServletRequest multipartRequest = (MultipartHttpServletRequest) request;
128     /** 构建图片保存的目录**/
129     String photoBookPathDir = "/upload";
130     /** 得到图片保存目录的真实路径**/
131     String photoBookRealPathDir = request.getSession().getServletContext().getRealPath(photoBookPathDir);
132     /** 根据真实路径创建目录**/
133     File photoBookSaveFile = new File(photoBookRealPathDir);
134     if(!photoBookSaveFile.exists())
135         photoBookSaveFile.mkdirs();
136     /** 页面控件的文件流**/
137     MultipartFile multipartFile_photoBook = multipartRequest.getFile(fileKeyName);
138     if(!multipartFile_photoBook.isEmpty()) {
139         /** 获取文件的后缀**/
140         String suffix = multipartFile_photoBook.getOriginalFilename().substring
141             (multipartFile_photoBook.getOriginalFilename().lastIndexOf( str "."));
142         /** 使用UUID生成文件名称**/
143         String photoBookFileName = UUID.randomUUID().toString()+ suffix; //构建文件名称
144         //String logImageName = multipartFile.getOriginalFilename();
145         /** 生成完整的文件保存路径加文件**/
146         String photoBookFilePath = photoBookRealPathDir + File.separator + photoBookFileName;
147
148         File photoBookFile = new File(photoBookFilePath);
149
150         multipartFile_photoBook.transferTo(photoBookFile);
151
152         fileName = "upload/" + photoBookFileName;
153     }
154     return fileName;
155 }
156
157 /**向客户端输出操作成功或失败信息*/
158 59 usages
159 @ public void writeJsonResponse(HttpServletResponse response,boolean success,String message)
160     throws IOException, JSONException {
161     response.setContentType("text/json;charset=UTF-8");
162     PrintWriter out = response.getWriter();
163     //将要被返回到客户端的对象
164     JSONObject json=new JSONObject();
165     json.accumulate("success", success);
166     json.accumulate("message", message);
167     out.println(json.toString());
168     out.flush();
169     out.close();
170 }
171
172 }
173

```

DepartmentController:

```

1 package com.chengxusheji.controller;
2
3 import java.io.IOException;
4 import java.io.OutputStream;
5 import java.io.PrintWriter;
6 import java.io.UnsupportedEncodingException;
7 import java.text.SimpleDateFormat;
8 import java.util.ArrayList;
9 import java.util.List;
10
11 import javax.annotation.Resource;
12 import javax.servlet.http.HttpServletRequest;
13 import javax.servlet.http.HttpServletResponse;
14 import javax.servlet.http.HttpSession;
15
16 import org.json.JSONArray;
17 import org.json.JSONException;
18 import org.json.JSONObject;
19 import org.springframework.stereotype.Controller;
20 import org.springframework.ui.Model;
21 import org.springframework.validation.BindingResult;
22 import org.springframework.validation.annotation.Validated;
23 import org.springframework.web.bind.WebDataBinder;
24 import org.springframework.web.bind.annotation.InitBinder;
25 import org.springframework.web.bind.annotation.ModelAttribute;
26 import org.springframework.web.bind.annotation.PathVariable;
27 import org.springframework.web.bind.annotation.RequestMapping;
28 import org.springframework.web.bind.annotation.RequestMethod;
29 import com.chengxusheji.utils.ExportExcelUtil;
30 import com.chengxusheji.utils.UserException;
31 import com.chengxusheji.service.DepartmentService;
32 import com.chengxusheji.po.Department;
33
34 //Department管理控制层
35 @Controller
36 @RequestMapping("/Department")

```

```

37 public class DepartmentController extends BaseController {
38
39     /*业务层对象*/
40     @Resource DepartmentService departmentService;
41
42     @InitBinder("department")
43     public void initBinderDepartment(WebDataBinder binder) { binder.setFieldDefaultPrefix("department."); }
44     /*跳转到添加Department视图*/
45     @RequestMapping(value = "/add", method = RequestMethod.GET)
46     public String add(Model model, HttpServletRequest request) throws Exception {
47         model.addAttribute(new Department());
48         return "Department_add";
49     }
50
51     /*客户端ajax方式提交添加科室信息*/
52     @RequestMapping(value = "/add", method = RequestMethod.POST)
53     public void add(@Validated Department department, BindingResult br,
54         Model model, HttpServletRequest request, HttpServletResponse response) throws Exception {
55         String message = "";
56         boolean success = false;
57         if (br.hasErrors()) {
58             message = "输入信息不符合要求!";
59             writeJsonResponse(response, success, message);
60             return;
61         }
62         departmentService.addDepartment(department);
63         message = "科室信息添加成功!";
64         success = true;
65         writeJsonResponse(response, success, message);
66     }
67
68     /*ajax方式按照查询条件分页查询科室信息*/
69     @RequestMapping(value = {"/list"}, method = {RequestMethod.GET, RequestMethod.POST})
70     public void list(String departmentName, String birthDate, Integer page, Integer rows, Model model, HttpServletRequest request, HttpServletResponse response) throws Exception {
71         if (page == null || page == 0) page = 1;
72     }
73 }

```

```

73         if (departmentName == null) departmentName = "";
74         if (birthDate == null) birthDate = "";
75         if (rows != 0) departmentService.setRows(rows);
76         List<Department> departmentList = departmentService.queryDepartment(departmentName, birthDate, page);
77         /*计算总的页数和总的记录数*/
78         departmentService.queryTotalPageAndRecordNumber(departmentName, birthDate);
79         /*获取到总的页数数目*/
80         int totalPage = departmentService.getTotalPage();
81         /*当前查询条件下总记录数*/
82         int recordNumber = departmentService.getRecordNumber();
83         response.setContentType("text/json;charset=UTF-8");
84         PrintWriter out = response.getWriter();
85         //将要被返回到客户端的对象
86         JSONObject jsonObj = new JSONObject();
87         jsonObj.accumulate("total", recordNumber);
88         JSONArray jsonArray = new JSONArray();
89         for (Department department : departmentList) {
90             JSONObject jsonDepartment = department.getJsonObject();
91             jsonArray.put(jsonDepartment);
92         }
93         jsonObj.accumulate("rows", jsonArray);
94         out.println(jsonObj.toString());
95         out.flush();
96         out.close();
97     }
98
99     /*ajax方式按照查询条件分页查询科室信息*/
100    @RequestMapping(value = {"/listAll"}, method = {RequestMethod.GET, RequestMethod.POST})
101    public void listAll(HttpServletRequest response) throws Exception {
102        List<Department> departmentList = departmentService.queryAllDepartment();
103        response.setContentType("text/json;charset=UTF-8");
104        PrintWriter out = response.getWriter();
105        JSONArray jsonArray = new JSONArray();
106        for (Department department : departmentList) {
107            JSONObject jsonDepartment = new JSONObject();

```

```

108            jsonDepartment.accumulate("departmentId", department.getDepartmentId());
109            jsonDepartment.accumulate("departmentName", department.getDepartmentName());
110            jsonArray.put(jsonDepartment);
111        }
112        out.println(jsonArray.toString());
113        out.flush();
114        out.close();
115    }
116
117    /*前台按照查询条件分页查询科室信息*/
118    @RequestMapping(value = {"/frontList"}, method = {RequestMethod.GET, RequestMethod.POST})
119    public String frontList(String departmentName, String birthDate, Integer currentPage, Model model, HttpServletRequest request) throws Exception {
120        if (currentPage == null || currentPage == 0) currentPage = 1;
121        if (departmentName == null) departmentName = "";
122        if (birthDate == null) birthDate = "";
123        List<Department> departmentList = departmentService.queryDepartment(departmentName, birthDate, currentPage);
124        /*计算总的页数和总的记录数*/
125        departmentService.queryTotalPageAndRecordNumber(departmentName, birthDate);
126        /*获取到总的页数数目*/
127        int totalPage = departmentService.getTotalPage();
128        /*当前查询条件下总记录数*/
129        int recordNumber = departmentService.getRecordNumber();
130        request.setAttribute("departmentList", departmentList);
131        request.setAttribute("totalPage", totalPage);
132        request.setAttribute("recordNumber", recordNumber);
133        request.setAttribute("currentPage", currentPage);
134        request.setAttribute("departmentName", departmentName);
135        request.setAttribute("birthDate", birthDate);
136        return "Department/department_frontquery_result";
137    }
138
139    /*前台查询Department信息*/
140    @RequestMapping(value = {"/{departmentId}/frontshow"}, method = RequestMethod.GET)
141    public String frontshow(@PathVariable Integer departmentId, Model model, HttpServletRequest request) throws Exception {
142        /*根据主键departmentId获取Department对象*/
143        Department department = departmentService.getDepartment(departmentId);

```

```

144         request.setAttribute("department", department);
145         return "Department/department_frontshow";
146     }
147 }
148
149 /*ajax方式显示科室信息修改jsp视图页*/
150 @RequestMapping(value="/{departmentId}/update",method=RequestMethod.GET)
151 public void update(@PathVariable Integer departmentId,Model model,HttpServletRequest request,HttpServletResponse response) throws Exception {
152     /*根据主键departmentId获取Department对象*/
153     Department department = departmentService.getDepartment(departmentId);
154
155     response.setContentType("text/json;charset=UTF-8");
156     PrintWriter out = response.getWriter();
157     //将要被返回到客户端的对象
158     JSONObject jsonDepartment = department.getJsonObject();
159     out.println(jsonDepartment.toString());
160     out.flush();
161     out.close();
162 }
163
164 /*ajax方式更新科室信息*/
165 @RequestMapping(value="/{departmentId}/update",method=RequestMethod.POST)
166 public void update(@Validated Department department, BindingResult br,
167     Model model, HttpServletRequest request,HttpServletResponse response) throws Exception {
168     String message = "";
169     boolean success = false;
170     if (br.hasErrors()) {
171         message = "输入的信息有错误!";
172         writeJsonResponse(response, success, message);
173         return;
174     }
175     try {
176         departmentService.updateDepartment(department);
177         message = "科室信息更新成功!";
178         success = true;
179         writeJsonResponse(response, success, message);
180     } catch (Exception e) {
181         writeJsonResponse(response, success, message);
182     } catch (Exception e) {
183         e.printStackTrace();
184         message = "科室信息更新失败!";
185         writeJsonResponse(response, success, message);
186     }
187 }
188
189 /*删除科室信息*/
190 @RequestMapping(value="/{departmentId}/delete",method=RequestMethod.GET)
191 public String delete(@PathVariable Integer departmentId,HttpServletRequest request) throws UnsupportedEncodingException {
192     try {
193         departmentService.deleteDepartment(departmentId);
194         request.setAttribute("message", "科室信息删除成功!");
195         return "message";
196     } catch (Exception e) {
197         e.printStackTrace();
198         request.setAttribute("error", "科室信息删除失败!");
199         return "error";
200     }
201 }
202
203 /*ajax方式删除多条科室信息记录*/
204 @RequestMapping(value="/deletes",method=RequestMethod.POST)
205 public void delete(String departmentIds,HttpServletRequest request,HttpServletResponse response) throws IOException, JSONException {
206     String message = "";
207     boolean success = false;
208     try {
209         int count = departmentService.deleteDepartments(departmentIds);
210         success = true;
211         message = count + "条记录删除成功";
212         writeJsonResponse(response, success, message);
213     } catch (Exception e) {
214         e.printStackTrace();
215         message = "有记录存在外键约束,删除失败";
216         writeJsonResponse(response, success, message);
217     }
218 }

```

```

217     }
218
219     /*按照查询条件导出科室信息信息到Excel*/
220     @RequestMapping(value = { "/OutToExcel" }, method = {RequestMethod.GET,RequestMethod.POST})
221     public void OutToExcel(String departmentName,String birthDate, Model model, HttpServletRequest request,HttpServletResponse response) th
222     {
223         if(departmentName == null) departmentName = "";
224         if(birthDate == null) birthDate = "";
225         List<Department> departmentList = departmentService.queryDepartment(departmentName,birthDate);
226         ExportExcelUtil ex = new ExportExcelUtil();
227         String _title = "Department信息记录";
228         String[] headers = { "科室id","科室名称","成立日期","负责人" };
229         List<String[]> dataset = new ArrayList<>();
230         for(int i=0;i<departmentList.size();i++) {
231             Department department = departmentList.get(i);
232             dataset.add(new String[]{department.getDepartmentId() + "", department.getDepartmentName(), department.getBirthDate(), department.
233         }
234         /*
235         OutputStream out = null;
236         try {
237             out = new FileOutputStream("C://output.xls");
238             ex.exportExcel(title,headers, dataset, out);
239             out.close();
240         } catch (Exception e) {
241             e.printStackTrace();
242         }
243         */
244         OutputStream out = null; //创建一个输出流对象
245
246         try {
247             out = response.getOutputStream(); //
248             response.setHeader( "Content-disposition", s1: "attachment; filename="+ "Department.xls"); //filename是下载的xls的名, 建议最好用英文
249             response.setContentType("application/msexcel;charset=UTF-8"); //设置类型
250             response.setHeader( "Pragma", s1: "No-cache"); //设置头
251             response.setHeader( "Cache-Control", s1: "no-cache"); //设置头
252             response.setDateHeader( "Expires", t: 0); //设置日期头
253             String rootPath = request.getSession().getServletContext().getRealPath( "/" );
254             ex.exportExcel(rootPath,_title,headers, dataset, out);
255             out.flush();
256         } catch (IOException e) {
257             e.printStackTrace();
258         } finally{
259             try{
260                 if(out!=null){
261                     out.close();
262                 }
263             }catch(IOException e){
264                 e.printStackTrace();
265             }
266         }
267     }

```

DoctorController:

```
DoctorController.java
1 package com.chengxusheji.controller;
2
3 import com.chengxusheji.po.Admin;
4 import com.chengxusheji.service.AdminService;
5 import java.io.IOException;
6 import java.io.OutputStream;
7 import java.io.PrintWriter;
8 import java.io.UnsupportedEncodingException;
9 import java.text.SimpleDateFormat;
10 import java.util.ArrayList;
11 import java.util.List;
12
13 import javax.annotation.Resource;
14 import javax.servlet.http.HttpServletRequest;
15 import javax.servlet.http.HttpServletResponse;
16 import javax.servlet.http.HttpSession;
17
18 import org.json.JSONArray;
19 import org.json.JSONException;
20 import org.json.JSONObject;
21 import org.springframework.stereotype.Controller;
22 import org.springframework.ui.Model;
23 import org.springframework.validation.BindingResult;
24 import org.springframework.validation.annotation.Validated;
25 import org.springframework.web.bind.WebDataBinder;
26 import org.springframework.web.bind.annotation.InitBinder;
27 import org.springframework.web.bind.annotation.ModelAttribute;
28 import org.springframework.web.bind.annotation.PathVariable;
29 import org.springframework.web.bind.annotation.RequestMapping;
30 import org.springframework.web.bind.annotation.RequestMethod;
31 import com.chengxusheji.utils.ExportExcelUtil;
32 import com.chengxusheji.utils.UserException;
33 import com.chengxusheji.service.DoctorService;
34 import com.chengxusheji.po.Doctor;
35 import com.chengxusheji.service.DepartmentService;
36 import com.chengxusheji.po.Department;
37
```

```

38 //Doctor管理控制层
39 @Controller
40 @RequestMapping("/Doctor")
41 public class DoctorController extends BaseController {
42
43     /*业务层对象*/
44     @Resource DoctorService doctorService;
45
46     @Resource DepartmentService departmentService;
47
48     @Resource
49     AdminService adminService;
50     @InitBinder("departmentObj")
51     public void initBinderdepartmentObj(WebDataBinder binder) { binder.setFieldDefaultPrefix("departmentObj."); }
52     @InitBinder("doctor")
53     public void initBinderDoctor(WebDataBinder binder) { binder.setFieldDefaultPrefix("doctor."); }
54
55     /*跳转到添加Doctor视图*/
56     @RequestMapping(value = "/add", method = RequestMethod.GET)
57     public String add(Model model, HttpServletRequest request) throws Exception {
58         model.addAttribute(new Doctor());
59         /*查询所有的Department信息*/
60         List<Department> departmentList = departmentService.queryAllDepartment();
61         request.setAttribute("departmentList", departmentList);
62         return "Doctor_add";
63     }
64
65     /*客户端ajax方式提交添加医生信息*/
66     @RequestMapping(value = "/add", method = RequestMethod.POST)
67     public void add(@Validated Doctor doctor, BindingResult br,
68         Model model, HttpServletRequest request, HttpServletResponse response) throws Exception {
69         String message = "";
70         boolean success = false;
71         if (br.hasErrors()) {
72             message = "输入信息不符合要求!";
73

```

```

74             writeJsonResponse(response, success, message);
75             return ;
76         }
77         if (doctorService.getDoctor(doctor.getDoctorNumber()) != null) {
78             message = "医生工号已经存在!";
79             writeJsonResponse(response, success, message);
80             return ;
81         }
82         Admin admin = adminService.findAdminByUserName(doctor.getDoctorNumber());
83         if (admin != null) {
84             message = "用户名已经存在!";
85             writeJsonResponse(response, success, message);
86             return ;
87         }
88         try {
89             doctor.setDoctorPhoto(this.handlePhotoUpload(request, fileKeyName: "doctorPhotoFile"));
90         } catch (UserException ex) {
91             message = "图片格式不正确!";
92             writeJsonResponse(response, success, message);
93             return ;
94         }
95         admin = new Admin();
96         admin.setPassword(doctor.getPassword());
97         admin.setUsername(doctor.getDoctorNumber());
98         admin.setType(1);
99         adminService.addAdmin(admin);
100         admin = adminService.findAdminByUserName(doctor.getDoctorNumber());
101         doctor.setAdminid(admin.getId());
102         doctorService.addDoctor(doctor);
103         message = "医生信息添加成功!";
104         success = true;
105         writeJsonResponse(response, success, message);
106     }
107
108     /*ajax方式按照查询条件分页查询医生信息*/
109

```

```

110 @RequestMapping(value = { @"/list" }, method = {RequestMethod.GET,RequestMethod.POST})
111 public void list(String doctorNumber,@ModelAttribute("departmentObj") Department departmentObj,String doctorName,String birthDate,Integer
112     if (page==null || page == 0) page = 1;
113     if (doctorNumber == null) doctorNumber = "";
114     if (doctorName == null) doctorName = "";
115     if (birthDate == null) birthDate = "";
116     if(rows != 0)doctorService.setRows(rows);
117     List<Doctor> doctorList = doctorService.queryDoctor(doctorNumber, departmentObj, doctorName, birthDate, page);
118     /*计算总的页数和总的记录数*/
119     doctorService.queryTotalPageAndRecordNumber(doctorNumber, departmentObj, doctorName, birthDate);
120     /*获取到总的页码数目*/
121     int totalPage = doctorService.getTotalPage();
122     /*当前查询条件下总记录数*/
123     int recordNumber = doctorService.getRecordNumber();
124     response.setContentType("text/json;charset=UTF-8");
125     PrintWriter out = response.getWriter();
126     //将要返回到客户端的对象
127     JSONObject jsonObj=new JSONObject();
128     jsonObj.accumulate("total", recordNumber);
129     JSONArray jsonArray = new JSONArray();
130     for(Doctor doctor:doctorList) {
131         JSONObject jsonDoctor = doctor.getJsonObject();
132         jsonArray.put(jsonDoctor);
133     }
134     jsonObj.accumulate("rows", jsonArray);
135     out.println(jsonObj.toString());
136     out.flush();
137     out.close();
138 }
139
140 /*ajax方式按照查询条件分页查询医生信息*/
141 @RequestMapping(value = { @"/listAll" }, method = {RequestMethod.GET,RequestMethod.POST})
142 public void listAll(HttpServletResponse response) throws Exception {
143     List<Doctor> doctorList = doctorService.queryAllDoctor();
144     response.setContentType("text/json;charset=UTF-8");
145     PrintWriter out = response.getWriter();
146     JSONArray jsonArray = new JSONArray();
147     for(Doctor doctor:doctorList) {
148         JSONObject jsonDoctor = new JSONObject();
149         jsonDoctor.accumulate("doctorNumber", doctor.getDoctorNumber());
150         jsonDoctor.accumulate("doctorName", doctor.getDoctorName());
151         jsonArray.put(jsonDoctor);
152     }
153     out.println(jsonArray.toString());
154     out.flush();
155     out.close();
156 }
157
158 /*前台按照查询条件分页查询医生信息*/
159 @RequestMapping(value = { @"/frontlist" }, method = {RequestMethod.GET,RequestMethod.POST})
160 public String frontlist(String doctorNumber,@ModelAttribute("departmentObj") Department departmentObj,String doctorName,String birthDate,Integer
161     if (currentPage==null || currentPage == 0) currentPage = 1;
162     if (doctorNumber == null) doctorNumber = "";
163     if (doctorName == null) doctorName = "";
164     if (birthDate == null) birthDate = "";
165     List<Doctor> doctorList = doctorService.queryDoctor(doctorNumber, departmentObj, doctorName, birthDate, currentPage);
166     /*计算总的页数和总的记录数*/
167     doctorService.queryTotalPageAndRecordNumber(doctorNumber, departmentObj, doctorName, birthDate);
168     /*获取到总的页码数目*/
169     int totalPage = doctorService.getTotalPage();
170     /*当前查询条件下总记录数*/
171     int recordNumber = doctorService.getRecordNumber();
172     request.setAttribute("doctorList", doctorList);
173     request.setAttribute("totalPage", totalPage);
174     request.setAttribute("recordNumber", recordNumber);
175     request.setAttribute("currentPage", currentPage);
176     request.setAttribute("doctorNumber", doctorNumber);
177     request.setAttribute("departmentObj", departmentObj);
178     request.setAttribute("doctorName", doctorName);
179     request.setAttribute("birthDate", birthDate);
180     List<Department> departmentList = departmentService.queryAllDepartment();
181     request.setAttribute("departmentList", departmentList);
182     return "Doctor/doctor_frontquery_result";

```

```

183     }
184
185     /*前台查询Doctor信息*/
186     @RequestMapping(value="/{doctorNumber}/frontshow",method=RequestMethod.GET)
187     public String frontshow(@PathVariable String doctorNumber,Model model,HttpServletRequest request) throws Exception {
188         /*根据主键doctorNumber获取Doctor对象*/
189         Doctor doctor = doctorService.getDoctor(doctorNumber);
190
191         List<Department> departmentList = departmentService.queryAllDepartment();
192         request.setAttribute("departmentList", departmentList);
193         request.setAttribute("doctor", doctor);
194         return "Doctor/doctor_frontshow";
195     }
196
197     /*ajax方式显示医生信息修改jsp视图页*/
198     @RequestMapping(value="/{doctorNumber}/update",method=RequestMethod.GET)
199     public void update(@PathVariable String doctorNumber,Model model,HttpServletRequest request,HttpServletResponse response) throws Exception {
200         /*根据主键doctorNumber获取Doctor对象*/
201         Doctor doctor = doctorService.getDoctor(doctorNumber);
202
203         response.setContentType("text/json;charset=UTF-8");
204         PrintWriter out = response.getWriter();
205         //将要被返回到客户端的对象
206         JSONObject jsonDoctor = doctor.getJsonObject();
207         out.println(jsonDoctor.toString());
208         out.flush();
209         out.close();
210     }
211
212     /*ajax方式更新医生信息*/
213     @RequestMapping(value="/{doctorNumber}/update", method = RequestMethod.POST)
214     public void update(@Validated Doctor doctor, BindingResult br,
215         Model model, HttpServletRequest request,HttpServletResponse response) throws Exception {
216         String message = "";
217         boolean success = false;
218         if (br.hasErrors()) {
219             message = "输入的信息有错误!";
220             writeJsonResponse(response, success, message);
221             return;
222         }
223         String doctorPhotoFileName = this.handlePhotoUpload(request, "doctorPhotoFile");
224         if(!doctorPhotoFileName.equals("upload/NoImage.jpg"))doctor.setDoctorPhoto(doctorPhotoFileName);
225
226
227         try {
228             doctorService.updateDoctor(doctor);
229             message = "医生信息更新成功!";
230             success = true;
231             writeJsonResponse(response, success, message);
232         } catch (Exception e) {
233             e.printStackTrace();
234             message = "医生信息更新失败!";
235             writeJsonResponse(response, success, message);
236         }
237     }
238
239     /*删除医生信息*/
240     @RequestMapping(value="/{doctorNumber}/delete",method=RequestMethod.GET)
241     public String delete(@PathVariable String doctorNumber,HttpServletRequest request) throws UnsupportedEncodingException {
242         try {
243             doctorService.deleteDoctor(doctorNumber);
244             request.setAttribute("message", "医生信息删除成功!");
245             return "message";
246         } catch (Exception e) {
247             e.printStackTrace();
248             request.setAttribute("error", "医生信息删除失败!");
249             return "error";
250         }
251     }
252 }

```

```

254      /*ajax方式删除多条医生信息记录*/
255      @RequestMapping(value="/delete",method=RequestMethod.POST)
256      public void delete(String doctorNumbers,HttpServletRequest request,HttpServletResponse response) throws IOException, JSONException {
257          String message = "";
258          boolean success = false;
259          try {
260              int count = doctorService.deleteDoctors(doctorNumbers);
261              success = true;
262              message = count + "条记录删除成功";
263              writeJsonResponse(response, success, message);
264          } catch (Exception e) {
265              //e.printStackTrace();
266              message = "有记录存在外键约束,删除失败";
267              writeJsonResponse(response, success, message);
268          }
269      }
270
271      /*按照查询条件导出医生信息信息到Excel*/
272      @RequestMapping(value = {"/","/OutToExcel"}, method = {RequestMethod.GET,RequestMethod.POST})
273      public void OutToExcel(String doctorNumber,@ModelAttribute("departmentObj") Department departmentObj,String doctorName,String birthDate,
274          if(doctorNumber == null) doctorNumber = "";
275          if(doctorName == null) doctorName = "";
276          if(birthDate == null) birthDate = "";
277          List<Doctor> doctorList = doctorService.queryDoctor(doctorNumber,departmentObj,doctorName,birthDate);
278          ExportExcelUtil ex = new ExportExcelUtil();
279          String _title = "Doctor信息记录";
280          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"};
281          List<String[]> dataset = new ArrayList<>();
282          for(int i=0;i<doctorList.size();i++) {
283              Doctor doctor = doctorList.get(i);
284              dataset.add(new String[]{doctor.getDoctorNumber(),doctor.getDepartmentObj().getDepartmentName(),doctor.getDoctorName(),doctor.ge
285          }
286          /*
287          OutputStream out = null;
288          try {
289              out = new FileOutputStream("C://output.xls");
290              ex.exportExcel(title,headers, dataset, out);
291              out.close();
292          } catch (Exception e) {
293              e.printStackTrace();
294          }
295          */
296          OutputStream out = null;//创建一个输出流对象
297          try {
298              out = response.getOutputStream();//
299              response.setHeader("Content-disposition", s1:"attachment; filename="+ "Doctor.xls");//filename是下载的xls的名,建议最好用英文
300              response.setContentType("application/msexcel;charset=UTF-8");//设置类型
301              response.setHeader("Pragma", s1:"No-cache");//设置头
302              response.setHeader("Cache-Control", s1:"no-cache");//设置头
303              response.setDateHeader("Expires", 0);//设置日期头
304              String rootPath = request.getSession().getServletContext().getRealPath(s1:"/");
305              ex.exportExcel(rootPath,_title,headers, dataset, out);
306              out.flush();
307          } catch (IOException e) {
308              e.printStackTrace();
309          }finally{
310              try{
311                  if(out!=null){
312                      out.close();
313                  }
314              }catch(IOException e){
315                  e.printStackTrace();
316              }
317          }
318      }
319  }
320

```

LeavewordController:

```

1  package com.chengxusheji.controller;
2
3  import java.io.IOException;
4  import java.io.OutputStream;
5  import java.io.PrintWriter;
6  import java.io.UnsupportedEncodingException;
7  import java.text.SimpleDateFormat;
8  import java.util.ArrayList;
9  import java.util.List;
10
11  import javax.annotation.Resource;
12  import javax.servlet.http.HttpServletRequest;
13  import javax.servlet.http.HttpServletResponse;
14  import javax.servlet.http.HttpSession;
15
16  import org.json.JSONArray;
17  import org.json.JSONException;
18  import org.json.JSONObject;
19  import org.springframework.stereotype.Controller;
20  import org.springframework.ui.Model;
21  import org.springframework.validation.BindingResult;
22  import org.springframework.validation.annotation.Validated;
23  import org.springframework.web.bind.WebDataBinder;
24  import org.springframework.web.bind.annotation.InitBinder;
25  import org.springframework.web.bind.annotation.ModelAttribute;
26  import org.springframework.web.bind.annotation.PathVariable;
27  import org.springframework.web.bind.annotation.RequestMapping;
28  import org.springframework.web.bind.annotation.RequestMethod;
29  import com.chengxusheji.utils.ExportExcelUtil;
30  import com.chengxusheji.utils.UserException;
31  import com.chengxusheji.service.LeavewordService;
32  import com.chengxusheji.po.Leaveword;
33  import com.chengxusheji.service.UserInfoService;
34  import com.chengxusheji.po.UserInfo;
35
36  //Leaveword管理控制层
37  @Controller

```

```

38  @RequestMapping(value = "/Leaveword")
39  public class LeavewordController extends BaseController {
40
41      /*业务层对象*/
42      @Resource LeavewordService leavewordService;
43
44      @Resource UserInfoService userInfoService;
45      @InitBinder("userObj")
46      public void initBinderuserObj(WebDataBinder binder) { binder.setFieldDefaultPrefix("userObj."); }
47      @InitBinder("leaveword")
48      public void initBinderLeaveword(WebDataBinder binder) { binder.setFieldDefaultPrefix("leaveword."); }
49
50      /*跳转到添加Leaveword页面*/
51      @RequestMapping(value = "/add", method = RequestMethod.GET)
52      public String add(Model model, HttpServletRequest request) throws Exception {
53          model.addAttribute(new Leaveword());
54          /*查询所有的UserInfo信息*/
55          List<UserInfo> userInfoList = userInfoService.queryAllUserInfo();
56          request.setAttribute("userInfoList", userInfoList);
57          return "Leaveword_add";
58      }
59
60      /*客户端ajax方式提交添加留言信息*/
61      @RequestMapping(value = "/add", method = RequestMethod.POST)
62      public void add(@Validated Leaveword leaveword, BindingResult br,
63          Model model, HttpServletRequest request, HttpServletResponse response, HttpSession session) throws Exception {
64          String message = "";
65          boolean success = false;
66          UserInfo userInfo = new UserInfo();
67          userInfo.setUser_name(session.getAttribute("username").toString());
68          leaveword.setUserObj(userInfo);
69          if (br.hasErrors()) {
70              message = "输入信息不符合要求! ";
71              writeJsonResponse(response, success, message);
72              return ;
73          }
74      }
75
76  }

```

```

77         leavewordService.addLeaveword(leaveword);
78         message = "留言添加成功!";
79         success = true;
80         writeJsonResponse(response, success, message);
81     }
82     /*ajax方式按照查询条件分页查询留言信息*/
83     @RequestMapping(value = { "/list" }, method = { RequestMethod.GET, RequestMethod.POST})
84     public void list(String leaveTitle, @ModelAttribute("userObj") UserInfo userObj, String leaveTime, Integer page, Integer rows, Model model) {
85         String type = session.getAttribute("type").toString();
86         String username = session.getAttribute("username").toString();
87         if("2".equals(type)){
88             userObj.setUser_name(username);
89         }
90         if (page==null || page == 0) page = 1;
91         if (leaveTitle == null) leaveTitle = "";
92         if (leaveTime == null) leaveTime = "";
93         if(rows != 0)leavewordService.setRows(rows);
94         List<Leaveword> leavewordList = leavewordService.queryLeaveword(leaveTitle, userObj, leaveTime, page);
95         /*计算总的页数和总的记录数*/
96         leavewordService.queryTotalPageAndRecordNumber(leaveTitle, userObj, leaveTime);
97         /*获取到总的页码数目*/
98         int totalPage = leavewordService.getTotalPage();
99         /*当前查询条件下总记录数*/
100         int recordNumber = leavewordService.getRecordNumber();
101         response.setContentType("text/json;charset=UTF-8");
102         PrintWriter out = response.getWriter();
103         //将要被返回到客户端的对象
104         JSONObject jsonObj=new JSONObject();
105         jsonObj.accumulate("total", recordNumber);
106         JSONArray jsonArray = new JSONArray();
107         for(Leaveword leaveword:leavewordList) {
108             JSONObject jsonLeaveword = leaveword.getJsonObject();
109             jsonArray.put(jsonLeaveword);
110         }
111         jsonObj.accumulate("rows", jsonArray);

```

```

112         out.println(jsonObj.toString());
113         out.flush();
114         out.close();
115     }
116
117     /*ajax方式按照查询条件分页查询留言信息*/
118     @RequestMapping(value = { "/listAll" }, method = { RequestMethod.GET, RequestMethod.POST})
119     public void listAll(HttpServletResponse response) throws Exception {
120         List<Leaveword> leavewordList = leavewordService.queryAllLeaveword();
121         response.setContentType("text/json;charset=UTF-8");
122         PrintWriter out = response.getWriter();
123         JSONArray jsonArray = new JSONArray();
124         for(Leaveword leaveword:leavewordList) {
125             JSONObject jsonLeaveword = new JSONObject();
126             jsonLeaveword.accumulate("leaveWordId", leaveword.getLeaveWordId());
127             jsonLeaveword.accumulate("leaveTitle", leaveword.getLeaveTitle());
128             jsonArray.put(jsonLeaveword);
129         }
130         out.println(jsonArray.toString());
131         out.flush();
132         out.close();
133     }
134
135     /*前台按照查询条件分页查询留言信息*/
136     @RequestMapping(value = { "/frontlist" }, method = { RequestMethod.GET, RequestMethod.POST})
137     public String frontlist(String leaveTitle, @ModelAttribute("userObj") UserInfo userObj, String leaveTime, Integer currentPage, Model model) {
138         String type = session.getAttribute("type").toString();
139         String username = session.getAttribute("username").toString();
140         if("2".equals(type)){
141             userObj.setUser_name(username);
142         }
143         if (currentPage==null || currentPage == 0) currentPage = 1;
144         if (leaveTitle == null) leaveTitle = "";
145         if (leaveTime == null) leaveTime = "";
146         List<Leaveword> leavewordList = leavewordService.queryLeaveword(leaveTitle, userObj, leaveTime, currentPage);
147         /*计算总的页数和总的记录数*/

```

```

148         leavewordService.queryTotalPageAndRecordNumber(leaveTitle, userObj, leaveTime);
149         /*获取到总的页码数目*/
150         int totalPage = leavewordService.getTotalPage();
151         /*当前查询条件下总记录数*/
152         int recordNumber = leavewordService.getRecordNumber();
153         request.setAttribute(Ⓢ "leavewordList", leavewordList);
154         request.setAttribute(Ⓢ "totalPage", totalPage);
155         request.setAttribute(Ⓢ "recordNumber", recordNumber);
156         request.setAttribute(Ⓢ "currentPage", currentPage);
157         request.setAttribute(Ⓢ "leaveTitle", leaveTitle);
158         request.setAttribute(Ⓢ "userObj", userObj);
159         request.setAttribute(Ⓢ "leaveTime", leaveTime);
160         List<UserInfo> userInfoList = userInfoService.queryAllUserInfo();
161         request.setAttribute(Ⓢ "userInfoList", userInfoList);
162         return "leaveword/leaveword_frontquery_result";
163     }
164
165     /*前台查询Leaveword信息*/
166     @RequestMapping(value=Ⓢ"/{leaveWordId}/frontshow",method=RequestMethod.GET)
167     public String frontshow(@PathVariable Integer leaveWordId,Model model,HttpServletRequest request) throws Exception {
168         /*根据主键leaveWordId获取Leaveword对象*/
169         Leaveword leaveword = leavewordService.getLeaveword(leaveWordId);
170
171         List<UserInfo> userInfoList = userInfoService.queryAllUserInfo();
172         request.setAttribute(Ⓢ "userInfoList", userInfoList);
173         request.setAttribute(Ⓢ "leaveword", leaveword);
174         return "leaveword/leaveword_frontshow";
175     }
176
177     /*ajax方式显示留言修改jsp视图页*/
178     @RequestMapping(value=Ⓢ"/{leaveWordId}/update",method=RequestMethod.GET)
179     public void update(@PathVariable Integer leaveWordId,Model model,HttpServletRequest request,HttpServletResponse response) throws Exception {
180         /*根据主键leaveWordId获取Leaveword对象*/
181         Leaveword leaveword = leavewordService.getLeaveword(leaveWordId);
182
183         response.setContentType("text/json;charset=UTF-8");
184         PrintWriter out = response.getWriter();

```

```

185         //将要返回到客户端的对象
186         JSONObject jsonLeaveword = leaveword.getJsonObject();
187         out.println(jsonLeaveword.toString());
188         out.flush();
189         out.close();
190     }
191
192     /*ajax方式更新留言信息*/
193     @RequestMapping(value=Ⓢ"/{leaveWordId}/update", method = RequestMethod.POST)
194     public void update(@Validated Leaveword leaveword, BindingResult br,
195         Model model, HttpServletRequest request,HttpServletResponse response) throws Exception {
196         String message = "";
197         boolean success = false;
198         if (br.hasErrors()) {
199             message = "输入的信息有错误! ";
200             writeJsonResponse(response, success, message);
201             return;
202         }
203         try {
204             leavewordService.updateLeaveword(leaveword);
205             message = "留言更新成功!";
206             success = true;
207             writeJsonResponse(response, success, message);
208         } catch (Exception e) {
209             e.printStackTrace();
210             message = "留言更新失败!";
211             writeJsonResponse(response, success, message);
212         }
213     }
214
215     /*删除留言信息*/
216     @RequestMapping(value=Ⓢ"/{leaveWordId}/delete",method=RequestMethod.GET)
217     public String delete(@PathVariable Integer leaveWordId,HttpServletRequest request) throws UnsupportedEncodingException {
218         try {
219             leavewordService.deleteLeaveword(leaveWordId);
220             request.setAttribute(Ⓢ "message", Ⓢ "留言删除成功!");

```

```

220         return "message";
221     } catch (Exception e) {
222         e.printStackTrace();
223         request.setAttribute("error", "留言删除失败!");
224         return "error";
225     }
226 }
227
228 }
229
230 /*ajax方式删除多条留言记录*/
231 @RequestMapping(value="/deletes",method=RequestMethod.POST)
232 public void delete(String leaveWordIds,HttpServletRequest request,HttpServletResponse response) throws IOException, JSONException {
233     String message = "";
234     boolean success = false;
235     try {
236         int count = leavewordService.deleteLeavewords(leaveWordIds);
237         success = true;
238         message = count + "条记录删除成功";
239         writeJsonResponse(response, success, message);
240     } catch (Exception e) {
241         //e.printStackTrace();
242         message = "有记录存在外键约束,删除失败";
243         writeJsonResponse(response, success, message);
244     }
245 }
246
247 /*按照查询条件导出留言信息到Excel*/
248 @RequestMapping(value = {"/OutToExcel"}, method = {RequestMethod.GET,RequestMethod.POST})
249 public void OutToExcel(String leaveTitle,@ModelAttribute("userObj") UserInfo userObj,String leaveTime, Model model, HttpServletRequest request) {
250     String type = session.getAttribute("type").toString();
251     String username = session.getAttribute("username").toString();
252     if("2".equals(type)){
253         userObj.setUser_name(username);
254     }
255     if(leaveTitle == null) leaveTitle = "";
256     if(leaveTime == null) leaveTime = "";

```

```

257     List<Leaveword> leavewordList = leavewordService.queryLeaveword(leaveTitle,userObj,leaveTime);
258     ExportExcelUtil ex = new ExportExcelUtil();
259     String _title = "Leaveword信息记录";
260     String[] headers = { "留言id","留言标题","留言人","留言时间","管理回复","回复时间";
261     List<String[]> dataset = new ArrayList<>();
262     for(int i=0;i<leavewordList.size();i++) {
263         Leaveword leaveword = leavewordList.get(i);
264         dataset.add(new String[]{leaveword.getLeaveWordId() + "",leaveword.getLeaveTitle(),leaveword.getUserObj().getName(),leaveword.getLeaveTime(),leaveword.getReplyTime()});
265     }
266     /*
267     OutputStream out = null;
268     try {
269         out = new FileOutputStream("C://output.xls");
270         ex.exportExcel(title,headers, dataset, out);
271         out.close();
272     } catch (Exception e) {
273         e.printStackTrace();
274     }
275     */
276     OutputStream out = null;//创建一个输出流对象
277     try {
278         out = response.getOutputStream();//
279         response.setHeader("Content-disposition", "attachment; filename="+_title+".xls");//filename是下载的文件名,建议最好用英文
280         response.setContentType("application/msexcel;charset=UTF-8");//设置类型
281         response.setHeader("Pragma", "No-cache");//设置头
282         response.setHeader("Cache-Control", "no-cache");//设置头
283         response.setDateHeader("Expires", 0);//设置日期头
284         String rootPath = request.getSession().getServletContext().getRealPath("/");
285         ex.exportExcel(rootPath,_title,headers, dataset, out);
286         out.flush();
287     } catch (IOException e) {
288         e.printStackTrace();
289     } finally{
290         try{
291             if(out!=null){

```

```

292                 out.close();
293             }
294         }catch(IOException e){
295             e.printStackTrace();
296         }
297     }
298 }
299 }
300

```

NewsController:

```
NewsController.java
1  package com.chengxusheji.controller;
2
3  import java.io.IOException;
4  import java.io.OutputStream;
5  import java.io.PrintWriter;
6  import java.io.UnsupportedEncodingException;
7  import java.text.SimpleDateFormat;
8  import java.util.ArrayList;
9  import java.util.List;
10
11  import javax.annotation.Resource;
12  import javax.servlet.http.HttpServletRequest;
13  import javax.servlet.http.HttpServletResponse;
14  import javax.servlet.http.HttpSession;
15
16  import org.json.JSONArray;
17  import org.json.JSONException;
18  import org.json.JSONObject;
19  import org.springframework.stereotype.Controller;
20  import org.springframework.ui.Model;
21  import org.springframework.validation.BindingResult;
22  import org.springframework.validation.annotation.Validated;
23  import org.springframework.web.bind.WebDataBinder;
24  import org.springframework.web.bind.annotation.InitBinder;
25  import org.springframework.web.bind.annotation.ModelAttribute;
26  import org.springframework.web.bind.annotation.PathVariable;
27  import org.springframework.web.bind.annotation.RequestMapping;
28  import org.springframework.web.bind.annotation.RequestMethod;
29  import com.chengxusheji.utils.ExportExcelUtil;
30  import com.chengxusheji.utils.UserException;
31  import com.chengxusheji.service.NewsService;
32  import com.chengxusheji.po.News;
33
34  //News管理控制层
35  @Controller
36  @RequestMapping("/News")
37  public class NewsController extends BaseController {
```

```

38
39      /*业务层对象*/
40      17 usages
41      @Resource NewsService newsService;
42
43      @InitBinder("news")
44      public void initBinderNews(WebDataBinder binder) { binder.setFieldDefaultPrefix("news."); }
45      /*跳转到添加News视图*/
46      @RequestMapping(value = {"/add"}, method = RequestMethod.GET)
47      public String add(Model model, HttpServletRequest request) throws Exception {
48          model.addAttribute(new News());
49          return "News_add";
50      }
51
52      /*客户端ajax方式提交添加新闻信息*/
53      @RequestMapping(value = {"/add"}, method = RequestMethod.POST)
54      public void add(@Validated News news, BindingResult br,
55          Model model, HttpServletRequest request, HttpServletResponse response) throws Exception {
56          String message = "";
57          boolean success = false;
58          if (br.hasErrors()) {
59              message = "输入信息不符合要求! ";
60              writeJsonResponse(response, success, message);
61              return ;
62          }
63          try {
64              news.setNewsPhoto(this.handlePhotoUpload(request, "newsPhotoFile"));
65          } catch (UserException ex) {
66              message = "图片格式不正确! ";
67              writeJsonResponse(response, success, message);
68              return ;
69          }
70          newsService.addNews(news);
71          message = "新闻信息添加成功!";
72          success = true;
73          writeJsonResponse(response, success, message);
74

```

```

75      }
76      /*ajax方式按照查询条件分页查询新闻信息*/
77      @RequestMapping(value = {"/list"}, method = {RequestMethod.GET, RequestMethod.POST})
78      public void list(String newsTitle, String newsDate, Integer page, Integer rows, Model model, HttpServletRequest request, HttpServletResponse response) throws Exception {
79          if (page == null || page == 0) page = 1;
80          if (newsTitle == null) newsTitle = "";
81          if (newsDate == null) newsDate = "";
82          if (rows != 0) newsService.setRows(rows);
83          List<News> newsList = newsService.queryNews(newsTitle, newsDate, page);
84          /*计算总的页数和总的记录数*/
85          newsService.queryTotalPageAndRecordNumber(newsTitle, newsDate);
86          /*获取到总的页码数目*/
87          int totalPage = newsService.getTotalPage();
88          /*当前查询条件下总记录数*/
89          int recordNumber = newsService.getRecordNumber();
90          response.setContentType("text/json;charset=UTF-8");
91          PrintWriter out = response.getWriter();
92          //将数据返回到客户端的对象
93          JSONObject jsonObj = new JSONObject();
94          jsonObj.accumulate("total", recordNumber);
95          JSONArray jsonArray = new JSONArray();
96          for (News news : newsList) {
97              JSONObject jsonNews = news.getJsonObject();
98              jsonArray.put(jsonNews);
99          }
100          jsonObj.accumulate("rows", jsonArray);
101          out.println(jsonObj.toString());
102          out.flush();
103          out.close();
104      }
105
106      /*ajax方式按照查询条件分页查询新闻信息*/
107      @RequestMapping(value = {"/listAll"}, method = {RequestMethod.GET, RequestMethod.POST})
108      public void listAll(HttpServletResponse response) throws Exception {
109          List<News> newsList = newsService.queryAllNews();
110          response.setContentType("text/json;charset=UTF-8");
111          PrintWriter out = response.getWriter();

```

```

112     JSONArray jsonArray = new JSONArray();
113     for (News news:newsList) {
114         JSONObject jsonNews = new JSONObject();
115         jsonNews.accumulate("newsId", news.getNewsId());
116         jsonNews.accumulate("newsTitle", news.getNewsTitle());
117         jsonArray.put(jsonNews);
118     }
119     out.println(jsonArray.toString());
120     out.flush();
121     out.close();
122 }
123
124 /*前台按照查询条件分页查询新闻信息*/
125 @RequestMapping(value = {"/frontlist"}, method = {RequestMethod.GET, RequestMethod.POST})
126 public String frontlist(String newsTitle, String newsDate, Integer currentPage, Model model, HttpServletRequest request) throws Exception {
127     if (currentPage == null || currentPage == 0) currentPage = 1;
128     if (newsTitle == null) newsTitle = "";
129     if (newsDate == null) newsDate = "";
130     List<News> newsList = newsService.queryNews(newsTitle, newsDate, currentPage);
131     /*计算总的页数和总的记录数*/
132     newsService.queryTotalPageAndRecordNumber(newsTitle, newsDate);
133     /*获取到总的页数*/
134     int totalPage = newsService.getTotalPage();
135     /*当前查询条件下总记录数*/
136     int recordNumber = newsService.getRecordNumber();
137     request.setAttribute("newsList", newsList);
138     request.setAttribute("totalPage", totalPage);
139     request.setAttribute("recordNumber", recordNumber);
140     request.setAttribute("currentPage", currentPage);
141     request.setAttribute("newsTitle", newsTitle);
142     request.setAttribute("newsDate", newsDate);
143     return "News/news_frontquery_result";
144 }
145
146 /*前台查询News信息*/
147 @RequestMapping(value = {"/{newsId}/frontshow"}, method = RequestMethod.GET)

```

```

148 public String frontshow(@PathVariable Integer newsId, Model model, HttpServletRequest request) throws Exception {
149     /*根据主键newsId获取News对象*/
150     News news = newsService.getNews(newsId);
151
152     request.setAttribute("news", news);
153     return "News/news_frontshow";
154 }
155
156 /*ajax方式显示新闻信息修改jsp视图页*/
157 @RequestMapping(value = {"/{newsId}/update"}, method = RequestMethod.GET)
158 public void update(@PathVariable Integer newsId, Model model, HttpServletRequest request, HttpServletResponse response) throws Exception {
159     /*根据主键newsId获取News对象*/
160     News news = newsService.getNews(newsId);
161
162     response.setContentType("text/json;charset=UTF-8");
163     PrintWriter out = response.getWriter();
164     //将要返回到客户端的对象
165     JSONObject jsonNews = news.getJsonObject();
166     out.println(jsonNews.toString());
167     out.flush();
168     out.close();
169 }
170
171 /*ajax方式更新新闻信息*/
172 @RequestMapping(value = {"/{newsId}/update"}, method = RequestMethod.POST)
173 public void update(@Validated News news, BindingResult br,
174     Model model, HttpServletRequest request, HttpServletResponse response) throws Exception {
175     String message = "";
176     boolean success = false;
177     if (br.hasErrors()) {
178         message = "输入的信息有错误!";
179         writeJsonResponse(response, success, message);
180     }
181     return;

```

```

181     }
182     String newsPhotoFileName = this.handlePhotoUpload(request, fileKeyName: "newsPhotoFile");
183     if(!newsPhotoFileName.equals("upload/NoImage.jpg"))news.setNewsPhoto(newsPhotoFileName);
184
185
186     try {
187         newsService.updateNews(news);
188         message = "新闻信息更新成功!";
189         success = true;
190         writeJsonResponse(response, success, message);
191     } catch (Exception e) {
192         e.printStackTrace();
193         message = "新闻信息更新失败!";
194         writeJsonResponse(response, success, message);
195     }
196 }
197
198 /*删除新闻信息信息*/
199 @RequestMapping(value="/{newsId}/delete",method=RequestMethod.GET)
200 public String delete(@PathVariable Integer newsId,HttpServletRequest request) throws UnsupportedEncodingException {
201     try {
202         newsService.deleteNews(newsId);
203         request.setAttribute( s: "message", o: "新闻信息删除成功!");
204         return "message";
205     } catch (Exception e) {
206         e.printStackTrace();
207         request.setAttribute( s: "error", o: "新闻信息删除失败!");
208         return "error";
209     }
210 }
211
212
213 /*ajax方式删除多条新闻信息记录*/
214 @RequestMapping(value="/deletes",method=RequestMethod.POST)
215 public void delete(String newsIds,HttpServletRequest request,HttpServletResponse response) throws IOException, JSONEx

```

```

216     String message = "";
217     boolean success = false;
218     try {
219         int count = newsService.deleteNews(newsIds);
220         success = true;
221         message = count + "条记录删除成功";
222         writeJsonResponse(response, success, message);
223     } catch (Exception e) {
224         //e.printStackTrace();
225         message = "有记录存在外键约束,删除失败";
226         writeJsonResponse(response, success, message);
227     }
228 }
229
230 /*按照查询条件导出新闻信息信息到Excel*/
231 @RequestMapping(value = { "/OutToExcel" }, method = {RequestMethod.GET,RequestMethod.POST})
232 public void OutToExcel(String newsTitle,String newsDate, Model model, HttpServletRequest request,HttpServletResponse response) throws IOException {
233     if(newsTitle == null) newsTitle = "";
234     if(newsDate == null) newsDate = "";
235     List<News> newsList = newsService.queryNews(newsTitle,newsDate);
236     ExportExcelUtil ex = new ExportExcelUtil();
237     String _title = "News信息记录";
238     String[] headers = { "新闻id","新闻标题","新闻图片","新闻日期","新闻来源"};
239     List<String[]> dataset = new ArrayList<>();
240     for(int i=0;i<newsList.size();i++) {
241         News news = newsList.get(i);
242         dataset.add(new String[]{news.getNewsId() + "",news.getNewsTitle(),news.getNewsPhoto(),news.getNewsDate(),news.getNewsFrom()});
243     }
244     /*
245     OutputStream out = null;
246     try {
247         out = new FileOutputStream("C://output.xls");
248         ex.exportExcel(title,headers, dataset, out);
249         out.close();
250     } catch (Exception e) {
251         e.printStackTrace();

```

```

252     }
253     */
254     OutputStream out = null; // 创建一个输出流对象
255     try {
256         out = response.getOutputStream(); //
257         response.setHeader("Content-disposition", "attachment; filename=" + "News.xls"); // filename是下载的文件名, 建议最好用英文
258         response.setContentType("application/msexcel; charset=UTF-8"); // 设置类型
259         response.setHeader("Pragma", "No-cache"); // 设置头
260         response.setHeader("Cache-Control", "no-cache"); // 设置头
261         response.setDateHeader("Expires", 0); // 设置日期头
262         String rootPath = request.getSession().getServletContext().getRealPath("/");
263         ex.exportExcel(rootPath, title, headers, dataset, out);
264         out.flush();
265     } catch (IOException e) {
266         e.printStackTrace();
267     } finally {
268         try {
269             if (out != null) {
270                 out.close();
271             }
272         } catch (IOException e) {
273             e.printStackTrace();
274         }
275     }
276 }
277 }
278

```

OrderInfoController:

```

OrderInfoController.java
1 package com.chengxusheji.controller;
2
3 import java.io.IOException;
4 import java.io.OutputStream;
5 import java.io.PrintWriter;
6 import java.io.UnsupportedEncodingException;
7 import java.text.SimpleDateFormat;
8 import java.util.ArrayList;
9 import java.util.List;
10
11 import javax.annotation.Resource;
12 import javax.servlet.http.HttpServletRequest;
13 import javax.servlet.http.HttpServletResponse;
14 import javax.servlet.http.HttpSession;
15
16 import org.json.JSONArray;
17 import org.json.JSONException;
18 import org.json.JSONObject;
19 import org.springframework.stereotype.Controller;
20 import org.springframework.ui.Model;
21 import org.springframework.validation.BindingResult;
22 import org.springframework.validation.annotation.Validated;
23 import org.springframework.web.bind.WebDataBinder;
24 import org.springframework.web.bind.annotation.InitBinder;
25 import org.springframework.web.bind.annotation.ModelAttribute;
26 import org.springframework.web.bind.annotation.PathVariable;
27 import org.springframework.web.bind.annotation.RequestMapping;
28 import org.springframework.web.bind.annotation.RequestMethod;
29 import com.chengxusheji.utils.ExportExcelUtil;
30 import com.chengxusheji.utils.UserException;
31 import com.chengxusheji.service.OrderInfoService;
32 import com.chengxusheji.po.OrderInfo;
33 import com.chengxusheji.service.DoctorService;
34 import com.chengxusheji.po.Doctor;
35 import com.chengxusheji.service.UserInfoService;
36 import com.chengxusheji.po.UserInfo;
37

```

```

38 //OrderInfo管理控制层
39 @Controller
40 @RequestMapping("/OrderInfo")
41 public class OrderInfoController extends BaseController {
42
43     /*业务层对象*/
44     @Resource OrderInfoService orderInfoService;
45
46     @Resource DoctorService doctorService;
47     @Resource UserInfoService userInfoService;
48     @InitBinder("userObj")
49     public void initBinderuserObj(WebDataBinder binder) { binder.setFieldDefaultPrefix("userObj."); }
50     @InitBinder("doctorObj")
51     public void initBinderdoctorObj(WebDataBinder binder) { binder.setFieldDefaultPrefix("doctorObj."); }
52     @InitBinder("orderInfo")
53     public void initBinderOrderInfo(WebDataBinder binder) { binder.setFieldDefaultPrefix("orderInfo."); }
54     /*跳转到添加OrderInfo视图*/
55     @RequestMapping(value = "/add", method = RequestMethod.GET)
56     public String add(Model model, HttpServletRequest request) throws Exception {
57         model.addAttribute(new OrderInfo());
58         /*查询所有的Doctor信息*/
59         List<Doctor> doctorList = doctorService.queryAllDoctor();
60         request.setAttribute("doctorList", doctorList);
61         /*查询所有的UserInfo信息*/
62         List<UserInfo> userInfoList = userInfoService.queryAllUserInfo();
63         request.setAttribute("userInfoList", userInfoList);
64         return "OrderInfo_add";
65     }
66
67     /*客户端ajax方式提交添加预约信息*/
68     @RequestMapping(value = "/add", method = RequestMethod.POST)
69     public void add(@Validated OrderInfo orderInfo, BindingResult br,
70         Model model, HttpServletRequest request, HttpServletResponse response) throws Exception {

```

```

71         String message = "";
72         boolean success = false;
73         if (br.hasErrors()) {
74             message = "输入信息不符合要求!";
75             writeJsonResponse(response, success, message);
76             return;
77         }
78         orderInfoService.addOrderInfo(orderInfo);
79         message = "预约信息添加成功!";
80         success = true;
81         writeJsonResponse(response, success, message);
82     }
83
84     /*ajax方式按照查询条件分页查询预约信息*/
85     @RequestMapping(value = "/list", method = {RequestMethod.GET, RequestMethod.POST})
86     public void list(@ModelAttribute("userObj") UserInfo userObj, @ModelAttribute("doctorObj") Doctor doctorObj, String orderDate, String timeInterval,
87         String type, String username) {
88         String type = session.getAttribute("type").toString();
89         String username = session.getAttribute("username").toString();
90         if ("2".equals(type)) {
91             userObj.setUser_name(username);
92         }
93         if (page == null || page == 0) page = 1;
94         if (orderDate == null) orderDate = "";
95         if (timeInterval == null) timeInterval = "";
96         if (telephone == null) telephone = "";
97         if (checkState == null) checkState = "";
98         if (orderTime == null) orderTime = "";
99         if (rows != 0) orderInfoService.setRows(rows);
100         List<OrderInfo> orderInfoList = orderInfoService.queryOrderInfo(userObj, doctorObj, orderDate, timeInterval, telephone, checkState, orderTime);
101         /*计算总的页数和总的记录数*/
102         orderInfoService.queryTotalPageAndRecordNumber(userObj, doctorObj, orderDate, timeInterval, telephone, checkState, orderTime);
103         /*获取到总的页数*/
104         int totalPage = orderInfoService.getTotalPage();
105         /*当前查询条件下总记录数*/
106         int recordNumber = orderInfoService.getRecordNumber();
107         response.setContentType("text/json;charset=UTF-8");
108         PrintWriter out = response.getWriter();
109         //将数据返回到客户端的页面

```

```

114         JSONObject jsonObj=new JSONObject();
115         jsonObj.accumulate("total", recordNumber);
116         JSONArray jsonArray = new JSONArray();
117         for(OrderInfo orderInfo:orderInfoList) {
118             JSONObject jsonObjOrderInfo = orderInfo.getJsonObject();
119             jsonArray.put(jsonObjOrderInfo);
120         }
121         jsonObj.accumulate("rows", jsonArray);
122         out.println(jsonObj.toString());
123         out.flush();
124         out.close();
125     }
126
127     /*ajax方式按照查询条件分页查询预约信息*/
128     @RequestMapping(value = {"/listAll"}, method = {RequestMethod.GET, RequestMethod.POST})
129     public void listAll(HttpServletRequest response) throws Exception {
130         List<OrderInfo> orderInfoList = orderInfoService.queryAllOrderInfo();
131         response.setContentType("text/json;charset=UTF-8");
132         PrintWriter out = response.getWriter();
133         JSONArray jsonArray = new JSONArray();
134         for(OrderInfo orderInfo:orderInfoList) {
135             JSONObject jsonObjOrderInfo = new JSONObject();
136             jsonObjOrderInfo.accumulate("orderId", orderInfo.getOrderId());
137             jsonArray.put(jsonObjOrderInfo);
138         }
139         out.println(jsonArray.toString());
140         out.flush();
141         out.close();
142     }
143
144     /*前台按照查询条件分页查询预约信息*/
145     @RequestMapping(value = {"/frontlist"}, method = {RequestMethod.GET, RequestMethod.POST})
146     public String frontlist(@ModelAttribute("userObj") UserInfo userObj, @ModelAttribute("doctorObj") Doctor doctorObj, String orderDate, String orderTime, String orderDateStr, String orderTimeStr, String telephone, String checkState, String orderTimeStr2) throws Exception {
147         String type = session.getAttribute("type").toString();
148         String username = session.getAttribute("username").toString();
149         if("2".equals(type)){
150             userObj.setUser_name(username);
151         }
152         if (currentPage==null || currentPage == 0) currentPage = 1;
153         if (orderDate == null) orderDate = "";
154         if (timeInterval == null) timeInterval = "";
155         if (telephone == null) telephone = "";
156         if (checkState == null) checkState = "";
157         if (orderTime == null) orderTime = "";
158         List<OrderInfo> orderInfoList = orderInfoService.queryOrderInfo(userObj, doctorObj, orderDate, timeInterval, telephone, checkState, orderTime);
159         /*计算总的页数和总的记录数*/
160         orderInfoService.queryTotalPageAndRecordNumber(userObj, doctorObj, orderDate, timeInterval, telephone, checkState, orderTime);
161         /*获取到总的页码数目*/
162         int totalPage = orderInfoService.getTotalPage();
163         /*当前查询条件下总记录数*/
164         int recordNumber = orderInfoService.getRecordNumber();
165         request.setAttribute("orderInfoList", orderInfoList);
166         request.setAttribute("totalPage", totalPage);
167         request.setAttribute("recordNumber", recordNumber);
168         request.setAttribute("currentPage", currentPage);
169         request.setAttribute("userObj", userObj);
170         request.setAttribute("doctorObj", doctorObj);
171         request.setAttribute("orderDate", orderDate);
172         request.setAttribute("timeInterval", timeInterval);
173         request.setAttribute("telephone", telephone);
174         request.setAttribute("checkState", checkState);
175         request.setAttribute("orderTime", orderTime);
176         List<Doctor> doctorList = doctorService.queryAllDoctor();
177         request.setAttribute("doctorList", doctorList);
178         List<UserInfo> userInfoList = userInfoService.queryAllUserInfo();
179         request.setAttribute("userInfoList", userInfoList);
180         return "OrderInfo/orderInfo_frontquery_result";
181     }
182
183     /*前台查询OrderInfo信息*/
184     @RequestMapping(value = {"/{orderId}/frontshow"}, method=RequestMethod.GET)
185     public String frontshow(@PathVariable Integer orderId, Model model, HttpServletRequest request) throws Exception {

```

```

186      /*根据主键orderId获取OrderInfo对象*/
187      OrderInfo orderInfo = orderInfoService.getOrderInfo(orderId);
188
189      List<Doctor> doctorList = doctorService.queryAllDoctor();
190      request.setAttribute( s: "doctorList", doctorList);
191      List<UserInfo> userInfoList = userInfoService.queryAllUserInfo();
192      request.setAttribute( s: "userInfoList", userInfoList);
193      request.setAttribute( s: "orderInfo", orderInfo);
194      return "OrderInfo/orderInfo_frontshow";
195  }
196
197  /*ajax方式显示预约信息修改jsp视图页*/
198  @RequestMapping(value="/{orderId}/update",method=RequestMethod.GET)
199  public void update(@PathVariable Integer orderId,Model model,HttpServletRequest request,HttpServletResponse response) throws Exception {
200      /*根据主键orderId获取OrderInfo对象*/
201      OrderInfo orderInfo = orderInfoService.getOrderInfo(orderId);
202
203      response.setContentType("text/json;charset=UTF-8");
204      PrintWriter out = response.getWriter();
205      //将要被返回到客户端的对象
206      JSONObject jsonOrderInfo = orderInfo.getJsonObject();
207      out.println(jsonOrderInfo.toString());
208      out.flush();
209      out.close();
210  }
211
212  /*ajax方式更新预约信息*/
213  @RequestMapping(value="/{orderId}/update", method = RequestMethod.POST)
214  public void update(@Validated OrderInfo orderInfo, BindingResult br,
215      Model model, HttpServletRequest request,HttpServletResponse response) throws Exception {
216      String message = "";
217      boolean success = false;
218      if (br.hasErrors()) {
219          message = "输入的信息有错误! ";
220          writeJsonResponse(response, success, message);

```

```

221      return;
222  }
223  try {
224      orderInfoService.updateOrderInfo(orderInfo);
225      message = "预约信息更新成功!";
226      success = true;
227      writeJsonResponse(response, success, message);
228  } catch (Exception e) {
229      e.printStackTrace();
230      message = "预约信息更新失败!";
231      writeJsonResponse(response, success, message);
232  }
233  }
234  /*删除预约信息*/
235  @RequestMapping(value="/{orderId}/delete",method=RequestMethod.GET)
236  public String delete(@PathVariable Integer orderId,HttpServletRequest request) throws UnsupportedEncodingException {
237      try {
238          orderInfoService.deleteOrderInfo(orderId);
239          request.setAttribute( s: "message", o: "预约信息删除成功!");
240          return "message";
241      } catch (Exception e) {
242          e.printStackTrace();
243          request.setAttribute( s: "error", o: "预约信息删除失败!");
244          return "error";
245      }
246  }
247  }
248  }
249
250  /*ajax方式删除多条预约信息记录*/
251  @RequestMapping(value="/{delete}",method=RequestMethod.POST)
252  public void delete(String orderIds,HttpServletRequest request,HttpServletResponse response) throws IOException, JSONException {
253      String message = "";
254      boolean success = false;
255      try {
256          int count = orderInfoService.deleteOrderInfos(orderIds);
257          success = true;

```

```

258         message = count + "条记录删除成功";
259         writeJsonResponse(response, success, message);
260     } catch (Exception e) {
261         //e.printStackTrace();
262         message = "有记录存在外键约束,删除失败";
263         writeJsonResponse(response, success, message);
264     }
265 }
266
267 /*按照查询条件导出预约信息到Excel*/
268 @RequestMapping(value = { @"/OutToExcel" }, method = {RequestMethod.GET,RequestMethod.POST})
269 public void OutToExcel(@ModelAttribute("userObj") UserInfo userObj,@ModelAttribute("doctorObj") Doctor doctorObj,String orderDate,String
270     if(orderDate == null) orderDate = "";
271     if(timeInterval == null) timeInterval = "";
272     if(telephone == null) telephone = "";
273     if(checkState == null) checkState = "";
274     if(orderTime == null) orderTime = "";
275     String type = session.getAttribute("type").toString();
276     String username = session.getAttribute("username").toString();
277     if("2".equals(type)){
278         userObj.setUser_name(username);
279     }
280     List<OrderInfo> orderInfoList = orderInfoService.queryOrderInfo(userObj,doctorObj,orderDate,timeInterval,telephone,checkState,orderT
281     ExportExcelUtil ex = new ExportExcelUtil();
282     String _title = "OrderInfo信息记录";
283     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"};
284     List<String[]> dataset = new ArrayList<>();
285     for(int i=0;i<orderInfoList.size();i++) {
286         OrderInfo orderInfo = orderInfoList.get(i);
287         dataset.add(new String[]{orderInfo.getOrderId() + "",orderInfo.getUserObj().getName(),orderInfo.getDoctorObj().getDoctorName(),
288     }
289     /*
290     OutputStream out = null;
291     try {
292         out = new FileOutputStream("C://output.xls");
293
294         ex.exportExcel(title,headers, dataset, out);
295         out.close();
296     } catch (Exception e) {
297         e.printStackTrace();
298     }
299     */
300     OutputStream out = null;//创建一个输出流对象
301     try {
302         out = response.getOutputStream();//
303         response.setHeader("Content-disposition", s1: "attachment; filename="+ "OrderInfo.xls");//filename是下载的xls的名, 建议最好用英文
304         response.setContentType("application/msexcel;charset=UTF-8");//设置类型
305         response.setHeader("Pragma", s1: "No-cache");//设置头
306         response.setHeader("Cache-Control", s1: "no-cache");//设置头
307         response.setDateHeader("Expires", 0);//设置日期头
308         String rootPath = request.getSession().getServletContext().getRealPath("/");
309         ex.exportExcel(rootPath,_title,headers, dataset, out);
310         out.flush();
311     } catch (IOException e) {
312         e.printStackTrace();
313     }finally{
314         try{
315             if(out!=null){
316                 out.close();
317             }
318         }catch(IOException e){
319             e.printStackTrace();
320         }
321     }
322 }
323

```

PatientController:

```

1 package com.chengxusheji.controller;
2
3 import java.io.IOException;
4 import java.io.OutputStream;
5 import java.io.PrintWriter;
6 import java.io.UnsupportedEncodingException;
7 import java.text.SimpleDateFormat;
8 import java.util.ArrayList;
9 import java.util.List;
10
11 import javax.annotation.Resource;
12 import javax.servlet.http.HttpServletRequest;
13 import javax.servlet.http.HttpServletResponse;
14 import javax.servlet.http.HttpSession;
15
16 import org.json.JSONArray;
17 import org.json.JSONException;
18 import org.json.JSONObject;
19 import org.springframework.stereotype.Controller;
20 import org.springframework.ui.Model;
21 import org.springframework.validation.BindingResult;
22 import org.springframework.validation.annotation.Validated;
23 import org.springframework.web.bind.WebDataBinder;
24 import org.springframework.web.bind.annotation.InitBinder;
25 import org.springframework.web.bind.annotation.ModelAttribute;
26 import org.springframework.web.bind.annotation.PathVariable;
27 import org.springframework.web.bind.annotation.RequestMapping;
28 import org.springframework.web.bind.annotation.RequestMethod;
29 import com.chengxusheji.utils.ExportExcelUtil;
30 import com.chengxusheji.utils.UserException;
31 import com.chengxusheji.service.PatientService;
32 import com.chengxusheji.po.Patient;
33 import com.chengxusheji.service.DoctorService;
34 import com.chengxusheji.po.Doctor;
35
36 //Patient管理控制层
37 @Controller

```

```

38 @RequestMapping(value = "/Patient")
39 public class PatientController extends BaseController {
40
41     /*业务层对象*/
42     @Resource PatientService patientService;
43
44     @Resource DoctorService doctorService;
45     @InitBinder("doctorObj")
46     public void initBinderdoctorObj(WebDataBinder binder) { binder.setFieldDefaultPrefix("doctorObj."); }
47     @InitBinder("patient")
48     public void initBinderPatient(WebDataBinder binder) { binder.setFieldDefaultPrefix("patient."); }
49
50     /*跳转到添加Patient视图*/
51     @RequestMapping(value = "/add", method = RequestMethod.GET)
52     public String add(Model model, HttpServletRequest request) throws Exception {
53         model.addAttribute(new Patient());
54         /*查询所有的Doctor信息*/
55         List<Doctor> doctorList = doctorService.queryAllDoctor();
56         request.setAttribute("doctorList", doctorList);
57         return "Patient_add";
58     }
59
60     /*客户端ajax方式提交添加病人信息*/
61     @RequestMapping(value = "/add", method = RequestMethod.POST)
62     public void add(@Validated Patient patient, BindingResult br,
63         Model model, HttpServletRequest request, HttpServletResponse response) throws Exception {
64         String message = "";
65         boolean success = false;
66         if (br.hasErrors()) {
67             message = "输入信息不符合要求! ";
68             writeJsonResponse(response, success, message);
69             return ;
70         }
71         patientService.addPatient(patient);
72         message = "病人信息添加成功!";
73         success = true;
74     }
75 }

```

```

77         writeJsonResponse(response, success, message);
78     }
79     /*ajax方式按照查询条件分页查询病人信息信息*/
80     @RequestMapping(value = { @RequestMapping("/list" }, method = {RequestMethod.GET,RequestMethod.POST})
81     public void list(@ModelAttribute("doctorObj") Doctor doctorObj,String patientName,String cardNumber,String telephone,String addTime,Integer page) {
82         if (page==null || page == 0) page = 1;
83         if (patientName == null) patientName = "";
84         if (cardNumber == null) cardNumber = "";
85         if (telephone == null) telephone = "";
86         if (addTime == null) addTime = "";
87         if(rows != 0)patientService.setRows(rows);
88         List<Patient> patientList = patientService.queryPatient(doctorObj, patientName, cardNumber, telephone, addTime, page);
89         /*计算总的页数和总的记录数*/
90         patientService.queryTotalPageAndRecordNumber(doctorObj, patientName, cardNumber, telephone, addTime);
91         /*获取到总的页码数目*/
92         int totalPage = patientService.getTotalPage();
93         /*当前查询条件下总记录数*/
94         int recordNumber = patientService.getRecordNumber();
95         response.setContentType("text/json;charset=UTF-8");
96         PrintWriter out = response.getWriter();
97         //将要被返回到客户端的对象
98         JSONObject jsonObj=new JSONObject();
99         jsonObj.accumulate("total", recordNumber);
100         JSONArray jsonArray = new JSONArray();
101         for(Patient patient:patientList) {
102             JSONObject jsonPatient = patient.getJsonObject();
103             jsonArray.put(jsonPatient);
104         }
105         jsonObj.accumulate("rows", jsonArray);
106         out.println(jsonObj.toString());
107         out.flush();
108         out.close();
109     }
110
111     /*ajax方式按照查询条件分页查询病人信息信息*/
112     @RequestMapping(value = { @RequestMapping("/listAll" }, method = {RequestMethod.GET,RequestMethod.POST})
113
114     public void listAll(HttpServletResponse response) throws Exception {
115         List<Patient> patientList = patientService.queryAllPatient();
116         response.setContentType("text/json;charset=UTF-8");
117         PrintWriter out = response.getWriter();
118         JSONArray jsonArray = new JSONArray();
119         for(Patient patient:patientList) {
120             JSONObject jsonPatient = new JSONObject();
121             jsonPatient.accumulate("patientId", patient.getPatientId());
122             jsonPatient.accumulate("patientName", patient.getPatientName());
123             jsonArray.put(jsonPatient);
124         }
125         out.println(jsonArray.toString());
126         out.flush();
127         out.close();
128     }
129
130     /*前台按照查询条件分页查询病人信息信息*/
131     @RequestMapping(value = { @RequestMapping("/frontlist" }, method = {RequestMethod.GET,RequestMethod.POST})
132     public String frontlist(@ModelAttribute("doctorObj") Doctor doctorObj,String patientName,String cardNumber,String telephone,String addTime,Integer currentPage) {
133         if (currentPage==null || currentPage == 0) currentPage = 1;
134         if (patientName == null) patientName = "";
135         if (cardNumber == null) cardNumber = "";
136         if (telephone == null) telephone = "";
137         if (addTime == null) addTime = "";
138         List<Patient> patientList = patientService.queryPatient(doctorObj, patientName, cardNumber, telephone, addTime, currentPage);
139         /*计算总的页数和总的记录数*/
140         patientService.queryTotalPageAndRecordNumber(doctorObj, patientName, cardNumber, telephone, addTime);
141         /*获取到总的页码数目*/
142         int totalPage = patientService.getTotalPage();
143         /*当前查询条件下总记录数*/
144         int recordNumber = patientService.getRecordNumber();
145         request.setAttribute("patientList", patientList);
146         request.setAttribute("totalPage", totalPage);
147         request.setAttribute("recordNumber", recordNumber);
148         request.setAttribute("currentPage", currentPage);
149         request.setAttribute("doctorObj", doctorObj);

```

```

149     request.setAttribute( "patientName", patientName);
150     request.setAttribute( "cardNumber", cardNumber);
151     request.setAttribute( "telephone", telephone);
152     request.setAttribute( "addTime", addTime);
153     List<Doctor> doctorList = doctorService.queryAllDoctor();
154     request.setAttribute( "doctorList", doctorList);
155     return "Patient/patient_frontquery_result";
156 }
157
158 /*前台查询Patient信息*/
159 @RequestMapping(value="/{patientId}/frontshow",method=RequestMethod.GET)
160 public String frontshow(@PathVariable Integer patientId,Model model,HttpServletRequest request) throws Exception {
161     /*根据主键patientId获取Patient对象*/
162     Patient patient = patientService.getPatient(patientId);
163
164     List<Doctor> doctorList = doctorService.queryAllDoctor();
165     request.setAttribute( "doctorList", doctorList);
166     request.setAttribute( "patient", patient);
167     return "Patient/patient_frontshow";
168 }
169
170 /*ajax方式显示病人信息修改jsp视图页*/
171 @RequestMapping(value="/{patientId}/update",method=RequestMethod.GET)
172 public void update(@PathVariable Integer patientId,Model model,HttpServletRequest request,HttpServletResponse response) throws Exception {
173     /*根据主键patientId获取Patient对象*/
174     Patient patient = patientService.getPatient(patientId);
175
176     response.setContentType("text/json;charset=UTF-8");
177     PrintWriter out = response.getWriter();
178     //将要被返回到客户端的对象
179     JSONObject jsonPatient = patient.getJsonObject();
180     out.println(jsonPatient.toString());
181     out.flush();

```

```

182     out.close();
183 }
184
185 /*ajax方式更新病人信息*/
186 @RequestMapping(value = "/{patientId}/update", method = RequestMethod.POST)
187 public void update(@Validated Patient patient, BindingResult br,
188     Model model, HttpServletRequest request,HttpServletResponse response) throws Exception {
189     String message = "";
190     boolean success = false;
191     if (br.hasErrors()) {
192         message = "输入的信息有错误!";
193         writeJsonResponse(response, success, message);
194         return;
195     }
196     try {
197         patientService.updatePatient(patient);
198         message = "病人信息更新成功!";
199         success = true;
200         writeJsonResponse(response, success, message);
201     } catch (Exception e) {
202         e.printStackTrace();
203         message = "病人信息更新失败!";
204         writeJsonResponse(response, success, message);
205     }
206 }
207
208 /*删除病人信息*/
209 @RequestMapping(value="/{patientId}/delete",method=RequestMethod.GET)
210 public String delete(@PathVariable Integer patientId,HttpServletRequest request) throws UnsupportedEncodingException {
211     try {
212         patientService.deletePatient(patientId);
213         request.setAttribute( "message", "病人信息删除成功!");
214         return "message";
215     } catch (Exception e) {
216         e.printStackTrace();
217         request.setAttribute( "error", "病人信息删除失败!");

```

```

217         return "error";
218     }
219 }
220
221 }
222
223 /*ajax方式删除多条病人信息记录*/
224 @RequestMapping(value="/delete",method=RequestMethod.POST)
225 public void delete(String patientIds,HttpServletRequest request,HttpServletResponse response) throws IOException, JSONException {
226     String message = "";
227     boolean success = false;
228     try {
229         int count = patientService.deletePatients(patientIds);
230         success = true;
231         message = count + "条记录删除成功";
232         writeJsonResponse(response, success, message);
233     } catch (Exception e) {
234         //e.printStackTrace();
235         message = "有记录存在外键约束,删除失败";
236         writeJsonResponse(response, success, message);
237     }
238 }
239
240 /*按照查询条件导出病人信息到Excel*/
241 @RequestMapping(value = {"/OutToExcel"}, method = {RequestMethod.GET,RequestMethod.POST})
242 public void OutToExcel(@ModelAttribute("doctorObj") Doctor doctorObj,String patientName,String cardNumber,String telephone,String addTime) {
243     if(patientName == null) patientName = "";
244     if(cardNumber == null) cardNumber = "";
245     if(telephone == null) telephone = "";
246     if(addTime == null) addTime = "";
247     List<Patient> patientList = patientService.queryPatient(doctorObj,patientName,cardNumber,telephone,addTime);
248     ExportExcelUtil ex = new ExportExcelUtil();
249     String _title = "Patient信息记录";
250
251     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"};
252     List<String[]> dataset = new ArrayList<>();
253     for(int i=0;i<patientList.size();i++) {
254         Patient patient = patientList.get(i);
255         dataset.add(new String[]{patient.getPatientId() + "",patient.getDoctorObj().getDoctorName(),patient.getPatientName(),patient.
256     }
257     /*
258     OutputStream out = null;
259     try {
260         out = new FileOutputStream("C://output.xls");
261         ex.exportExcel(title,headers,dataset, out);
262         out.close();
263     } catch (Exception e) {
264         e.printStackTrace();
265     }
266     */
267
268     OutputStream out = null;//创建一个输出流对象
269     try {
270         out = response.getOutputStream();//
271         response.setHeader("Content-disposition", s1: "attachment; filename="+ "Patient.xls");//filename是下载的文件名,建议最好用英文
272         response.setContentType("application/msexcel;charset=UTF-8");//设置类型
273         response.setHeader("Pragma", s1: "No-cache");//设置头
274         response.setHeader("Cache-Control", s1: "no-cache");//设置头
275         response.setDateHeader("Expires", 0);//设置过期头
276         String rootPath = request.getSession().getServletContext().getRealPath("/");
277         ex.exportExcel(rootPath,_title,headers,dataset, out);
278         out.flush();
279     } catch (IOException e) {
280         e.printStackTrace();
281     } finally{
282         try{
283             if(out!=null){
284                 out.close();
285             }
286         }catch(IOException e){
287             e.printStackTrace();
288         }
289     }
290 }

```

SystemController:

```

1 package com.chengxusheji.controller;
2
3 import com.chengxusheji.po.UserInfo;
4 import com.chengxusheji.service.UserInfoService;
5 import java.io.PrintWriter;
6
7 import javax.annotation.Resource;
8 import javax.servlet.http.HttpServletRequest;
9 import javax.servlet.http.HttpServletResponse;
10 import javax.servlet.http.HttpSession;
11
12 import org.json.JSONObject;
13 import org.springframework.stereotype.Controller;
14 import org.springframework.ui.Model;
15 import org.springframework.validation.BindingResult;
16 import org.springframework.validation.annotation.Validated;
17 import org.springframework.web.bind.annotation.RequestMapping;
18 import org.springframework.web.bind.annotation.RequestMethod;
19 import org.springframework.web.bind.annotation.RequestParam;
20 import org.springframework.web.bind.annotation.SessionAttributes;
21
22
23 import com.chengxusheji.po.Admin;
24 import com.chengxusheji.service.AdminService;
25 import com.chengxusheji.utils.UserException;
26
27
28 @Controller
29 @SessionAttributes("username")
30 public class SystemController extends BaseController{
31
32     7 usages
33     @Resource AdminService adminService;
34     no usages
35     @Resource
36     UserInfoService userInfoService;
37
38     @RequestMapping(value="/login",method=RequestMethod.GET)
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

```

73 public void login(@Validated Admin admin, BindingResult br, Model model, HttpServletRequest request, HttpServletResponse response, HttpSession session) {
74     boolean success = true;
75     String msg = "";
76     if (br.hasErrors()) {
77         msg = br.getAllErrors().toString();
78         success = false;
79     }
80     Admin user = adminService.checkLogin(admin);
81     if (user == null) {
82         msg = adminService.getErrMsg();
83         success = false;
84     }
85     if (success) {
86         session.setAttribute("username", user.getUsername());
87         session.setAttribute("type", user.getType());
88     }
89     response.setContentType("text/json;charset=UTF-8");
90     PrintWriter out = response.getWriter();
91     // 将要返回到客户端的对象
92     JSONObject json = new JSONObject();
93     json.accumulate("success", success);
94     json.accumulate("msg", msg);
95     out.println(json.toString());
96     out.flush();
97     out.close();
98 }
99
100
101
102 @RequestMapping("/logout")
103 public String logout(Model model, HttpSession session) {
104     model.asMap().remove("username");
105     session.invalidate();
106     return "redirect:/";
107 }

```

```

108
109
110 @RequestMapping(value = "/changePassword", method = RequestMethod.POST)
111 public String ChangePassword(String oldPassword, String newPassword, String newPassword2, HttpServletRequest request, HttpSession session) {
112     if (oldPassword.equals("")) throw new UserException("请输入旧密码!");
113     if (newPassword.equals("")) throw new UserException("请输入新密码!");
114     if (!newPassword.equals(newPassword2)) throw new UserException("两次新密码输入不一致!");
115
116     String username = (String) session.getAttribute("username");
117     if (username == null) throw new UserException("session会话超时, 请重新登录系统!");
118
119     Admin admin = adminService.findAdminByUsername(username);
120     if (!admin.getPassword().equals(oldPassword)) throw new UserException("输入的旧密码不正确!");
121
122     try {
123         adminService.changePassword(username, newPassword);
124         request.setAttribute("message", java.net.URLEncoder.encode("密码修改成功!", "GBK"));
125         return "message";
126     } catch (Exception e) {
127         e.printStackTrace();
128         request.setAttribute("error", java.net.URLEncoder.encode("密码修改失败!", "GBK"));
129         return "error";
130     }
131 }
132
133
134
135
136 @RequestMapping(value = "/register", method = RequestMethod.POST)
137 public void register(@Validated Admin admin, BindingResult br, Model model, HttpServletRequest request, HttpServletResponse response) throws Exception {
138     String message = "";
139     boolean success = false;
140     if (br.hasErrors()) {
141         message = "输入信息不符合要求! ";
142     }
143     writeJsonResponse(response, success, message);

```

```

144         return ;
145     }
146     if(adminService.findAdminByUserName(admin.getUsername()) != null) {
147         message = "用户名已经存在!";
148         writeJsonResponse(response, success, message);
149         return ;
150     }
151     adminService.addAdmin(admin);
152     message = "用户添加成功!";
153     success = true;
154     writeJsonResponse(response, success, message);
155 }
156
157
158 }
159

```

UserInfoController:

```

UserInfoController.java
1  package com.chengxusheji.controller;
2
3  import com.chengxusheji.po.Admin;
4  import com.chengxusheji.service.AdminService;
5  import java.io.IOException;
6  import java.io.OutputStream;
7  import java.io.PrintWriter;
8  import java.io.UnsupportedEncodingException;
9  import java.text.SimpleDateFormat;
10 import java.util.ArrayList;
11 import java.util.List;
12
13 import javax.annotation.Resource;
14 import javax.servlet.http.HttpServletRequest;
15 import javax.servlet.http.HttpServletResponse;
16 import javax.servlet.http.HttpSession;
17
18 import org.json.JSONArray;
19 import org.json.JSONException;
20 import org.json.JSONObject;
21 import org.springframework.stereotype.Controller;
22 import org.springframework.ui.Model;
23 import org.springframework.validation.BindingResult;
24 import org.springframework.validation.annotation.Validated;
25 import org.springframework.web.bind.WebDataBinder;
26 import org.springframework.web.bind.annotation.InitBinder;
27 import org.springframework.web.bind.annotation.ModelAttribute;
28 import org.springframework.web.bind.annotation.PathVariable;
29 import org.springframework.web.bind.annotation.RequestMapping;
30 import org.springframework.web.bind.annotation.RequestMethod;
31 import com.chengxusheji.utils.ExportExcelUtil;
32 import com.chengxusheji.utils.UserException;
33 import com.chengxusheji.service.UserInfoService;
34 import com.chengxusheji.po.UserInfo;
35
36 //UserInfo管理控制层
37 @Controller
38 @RequestMapping("/UserInfo")

```

```

38 @RequestMapping(value = "/UserInfo")
39 public class UserInfoController extends BaseController {
40
41     3 usages
42     @Resource
43     AdminService adminService;
44     /*业务层对象*/
45     18 usages
46     @Resource UserInfoService userInfoService;
47
48     @InitBinder("userInfo")
49     public void initBinderUserInfo(WebDataBinder binder) { binder.setFieldDefaultPrefix("userInfo."); }
50     /*跳转到添加UserInfo视图*/
51     @RequestMapping(value = "/add", method = RequestMethod.GET)
52     public String add(Model model, HttpServletRequest request) throws Exception {
53         model.addAttribute(new UserInfo());
54         return "UserInfo_add";
55     }
56
57     /*客户端ajax方式提交添加用户信息*/
58     @RequestMapping(value = "/add", method = RequestMethod.POST)
59     public void add(@Validated UserInfo userInfo, BindingResult br,
60         Model model, HttpServletRequest request, HttpServletResponse response) throws Exception {
61         String message = "";
62         boolean success = false;
63         if (br.hasErrors()) {
64             message = "输入信息不符合要求! ";
65             writeJsonResponse(response, success, message);
66             return ;
67         }
68         if(userInfoService.getUserInfo(userInfo.getUser_name()) != null) {
69             message = "用户名已经存在! ";
70             writeJsonResponse(response, success, message);
71             return ;
72         }
73         Admin admin = adminService.findAdminByUserName(userInfo.getUser_name());
74         if(admin != null) {

```

```

75             message = "用户名已经存在! ";
76             writeJsonResponse(response, success, message);
77             return ;
78         }
79         try {
80             userInfo.setUserPhoto(this.handlePhotoUpload(request, "userPhotoFile"));
81         } catch (UserException ex) {
82             message = "图片格式不正确! ";
83             writeJsonResponse(response, success, message);
84             return ;
85         }
86         admin = new Admin();
87         admin.setPassword(userInfo.getPassword());
88         admin.setUsername(userInfo.getUser_name());
89         admin.setType(2);
90         adminService.addAdmin(admin);
91         admin = adminService.findAdminByUserName(userInfo.getUser_name());
92         userInfo.setAdminid(admin.getId());
93         userInfoService.addUserInfo(userInfo);
94         message = "用户添加成功! ";
95         success = true;
96         writeJsonResponse(response, success, message);
97     }
98     /*ajax方式按照查询条件分页查询用户信息*/
99     @RequestMapping(value = {"/list"}, method = {RequestMethod.GET, RequestMethod.POST})
100     public void list(String user_name, String name, String birthDate, String telephone, Integer page, Integer rows, Model model, HttpServletRequest request) {
101         if (page == null || page == 0) page = 1;
102         if (user_name == null) user_name = "";
103         if (name == null) name = "";
104         if (birthDate == null) birthDate = "";
105         if (telephone == null) telephone = "";
106         if (rows != 0) userInfoService.setRows(rows);
107         List<UserInfo> userInfoList = userInfoService.queryUserInfo(user_name, name, birthDate, telephone, page);
108         /*计算总的页数和总的记录数*/
109         userInfoService.queryTotalPageAndRecordNumber(user_name, name, birthDate, telephone);

```

```

110      /*获取到总的页码数目*/
111      int totalPage = userInfoService.getTotalPage();
112      /*当前查询条件下总记录数*/
113      int recordNumber = userInfoService.getRecordNumber();
114      response.setContentType("text/json;charset=UTF-8");
115      PrintWriter out = response.getWriter();
116      //将要被返回到客户端的对象
117      JSONObject jsonObj=new JSONObject();
118      jsonObj.accumulate("total", recordNumber);
119      JSONArray jsonArray = new JSONArray();
120      for(UserInfo userInfo:userInfoList) {
121          JSONObject jsonUserInfo = userInfo.getJsonObject();
122          jsonArray.put(jsonUserInfo);
123      }
124      jsonObj.accumulate("rows", jsonArray);
125      out.println(jsonObj.toString());
126      out.flush();
127      out.close();
128  }
129
130  /*ajax方式按照查询条件分页查询用户信息*/
131  @RequestMapping(value = {"/listAll"}, method = {RequestMethod.GET,RequestMethod.POST})
132  public void listAll(HttpServletResponse response) throws Exception {
133      List<UserInfo> userInfoList = userInfoService.queryAllUserInfo();
134      response.setContentType("text/json;charset=UTF-8");
135      PrintWriter out = response.getWriter();
136      JSONArray jsonArray = new JSONArray();
137      for(UserInfo userInfo:userInfoList) {
138          JSONObject jsonUserInfo = new JSONObject();
139          jsonUserInfo.accumulate("user_name", userInfo.getUser_name());
140          jsonUserInfo.accumulate("name", userInfo.getName());
141          jsonArray.put(jsonUserInfo);
142      }
143      out.println(jsonArray.toString());

```

```

144      out.flush();
145      out.close();
146  }
147
148  /*前台按照查询条件分页查询用户信息*/
149  @RequestMapping(value = {"/frontlist"}, method = {RequestMethod.GET,RequestMethod.POST})
150  public String frontlist(String user_name,String name,String birthDate,String telephone,Integer currentPage, Model model, HttpServletRequest request) throws Exception {
151      if (currentPage==null || currentPage == 0) currentPage = 1;
152      if (user_name == null) user_name = "";
153      if (name == null) name = "";
154      if (birthDate == null) birthDate = "";
155      if (telephone == null) telephone = "";
156      List<UserInfo> userInfoList = userInfoService.queryUserInfo(user_name, name, birthDate, telephone, currentPage);
157      /*计算总的页数和总的记录数*/
158      userInfoService.queryTotalPageAndRecordNumber(user_name, name, birthDate, telephone);
159      /*获取到总的页码数目*/
160      int totalPage = userInfoService.getTotalPage();
161      /*当前查询条件下总记录数*/
162      int recordNumber = userInfoService.getRecordNumber();
163      request.setAttribute("userInfoList", userInfoList);
164      request.setAttribute("totalPage", totalPage);
165      request.setAttribute("recordNumber", recordNumber);
166      request.setAttribute("currentPage", currentPage);
167      request.setAttribute("user_name", user_name);
168      request.setAttribute("name", name);
169      request.setAttribute("birthDate", birthDate);
170      request.setAttribute("telephone", telephone);
171      return "UserInfo/userInfo_frontquery_result";
172  }
173
174  /*前台查询UserInfo信息*/
175  @RequestMapping(value={"/user_name"/frontshow"},method=RequestMethod.GET)
176  public String frontshow(@PathVariable String user_name,Model model,HttpServletRequest request) throws Exception {
177      /*根据主键user_name获取UserInfo对象*/
178      UserInfo userInfo = userInfoService.getUserInfo(user_name);

```

```

179
180     request.setAttribute("userInfo", userInfo);
181     return "UserInfo/userInfo_frontshow";
182 }
183
184 /*ajax方式显示用户修改jsp视图页*/
185 @RequestMapping(value="/{user_name}/update",method=RequestMethod.GET)
186 public void update(@PathVariable String user_name,Model model,HttpServletRequest request,HttpServletResponse response) throws Exception
187 {
188     /*根据主键user_name获取UserInfo对象*/
189     UserInfo userInfo = userInfoService.getUserInfo(user_name);
190
191     response.setContentType("text/json;charset=UTF-8");
192     PrintWriter out = response.getWriter();
193     //将要被返回到客户端的对象
194     JSONObject jsonUserInfo = userInfo.getJsonObject();
195     out.println(jsonUserInfo.toString());
196     out.flush();
197     out.close();
198 }
199
200 /*ajax方式更新用户信息*/
201 @RequestMapping(value="/{user_name}/update", method = RequestMethod.POST)
202 public void update(@Validated UserInfo userInfo, BindingResult br,
203     Model model, HttpServletRequest request,HttpServletResponse response) throws Exception {
204     String message = "";
205     boolean success = false;
206     if (br.hasErrors()) {
207         message = "输入的信息有错误! ";
208     }
209 }

```

```

207         writeJsonResponse(response, success, message);
208         return;
209     }
210     String userPhotoFileName = this.handlePhotoUpload(request, fileKeyName: "userPhotoFile");
211     if(!userPhotoFileName.equals("upload/NoImage.jpg"))userInfo.setUserPhoto(userPhotoFileName);
212
213
214     try {
215         userInfoService.updateUserInfo(userInfo);
216         message = "用户更新成功!";
217         success = true;
218         writeJsonResponse(response, success, message);
219     } catch (Exception e) {
220         e.printStackTrace();
221         message = "用户更新失败!";
222         writeJsonResponse(response, success, message);
223     }
224 }
225
226 /*删除用户信息*/
227 @RequestMapping(value="/{user_name}/delete",method=RequestMethod.GET)
228 public String delete(@PathVariable String user_name,HttpServletRequest request) throws UnsupportedEncodingException {
229     try {
230         userInfoService.deleteUserInfo(user_name);
231         request.setAttribute("message", "用户删除成功!");
232         return "message";
233     } catch (Exception e) {
234         e.printStackTrace();
235         request.setAttribute("error", "用户删除失败!");
236         return "error";
237     }
238 }
239 }

```

```

241  /*ajax方式删除多条用户记录*/
242  @RequestMapping(value="/deletes",method=RequestMethod.POST)
243  public void delete(String user_names,HttpServletRequest request,HttpServletResponse response) throws IOException, JSONException {
244      String message = "";
245      boolean success = false;
246      try {
247          int count = userInfoService.deleteUserInfos(user_names);
248          success = true;
249          message = count + "条记录删除成功";
250          writeJsonResponse(response, success, message);
251      } catch (Exception e) {
252          //e.printStackTrace();
253          message = "有记录存在外键约束,删除失败";
254          writeJsonResponse(response, success, message);
255      }
256  }
257
258  /*按照查询条件导出用户信息到Excel*/
259  @RequestMapping(value = {"/OutToExcel"}, method = {RequestMethod.GET,RequestMethod.POST})
260  public void OutToExcel(String user_name,String name,String birthDate,String telephone, Model model, HttpServletRequest request,HttpServlet
261      if(user_name == null) user_name = "";
262      if(name == null) name = "";
263      if(birthDate == null) birthDate = "";
264      if(telephone == null) telephone = "";
265      List<UserInfo> userInfoList = userInfoService.queryUserInfo(user_name,name,birthDate,telephone);
266      ExportExcelUtil ex = new ExportExcelUtil();
267      String _title = "UserInfo信息记录";
268      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"};
269      List<String[]> dataset = new ArrayList<>();
270      for(int i=0;i<userInfoList.size();i++) {
271          UserInfo userInfo = userInfoList.get(i);
272          dataset.add(new String[]{userInfo.getUser_name(),userInfo.getName(),userInfo.getGender(),userInfo.getBirthDate(),userInfo.getUse
273      }
274      /*
275      OutputStream out = null;
276      try {
277          out = new FileOutputStream("C://output.xls");
278          ex.exportExcel(_title,headers, dataset, out);
279          out.close();
280      } catch (Exception e) {
281          e.printStackTrace();
282      }
283      */
284      OutputStream out = null; //创建一个输出流对象
285      try {
286          out = response.getOutputStream(); //
287          response.setHeader("Content-disposition", s1: "attachment; filename="+ "UserInfo.xls"); //filename是下载的xls的名。建议最好用英文
288          response.setContentType("application/msexcel; charset=UTF-8"); //设置类型
289          response.setHeader("Pragma", s1: "No-cache"); //设置头
290          response.setHeader("Cache-Control", s1: "no-cache"); //设置头
291          response.setDateHeader("Expires", 1: 0); //设置日期头
292          String rootPath = request.getSession().getServletContext().getRealPath(s1: "/");
293          ex.exportExcel(rootPath,_title,headers, dataset, out);
294          out.flush();
295      } catch (IOException e) {
296          e.printStackTrace();
297      } finally{
298          try{
299              if(out!=null){
300                  out.close();
301              }
302          }catch(IOException e){
303              e.printStackTrace();
304          }
305      }
306  }
307  }
308

```

mapper 包:

AdminMapper:

```

1  package com.chengxusheji.mapper;
2
3
4  import com.chengxusheji.po.Admin;
5
6  3 usages
7  public interface AdminMapper {
8
9      2 usages
10     public Admin findAdminByUserName(String username) throws Exception;
11
12     1 usage
13     public void changePassword(Admin admin) throws Exception;
14
15     1 usage
16     public void addAdmin(Admin admin) throws Exception;
17 }

```

DepartmentMapper:

```

1  package com.chengxusheji.mapper;
2
3  import java.util.ArrayList;
4  import org.apache.ibatis.annotations.Param;
5  import com.chengxusheji.po.Department;
6
7  4 usages
8  public interface DepartmentMapper {
9      /*添加科室信息*/
10     1 usage
11     public void addDepartment(Department department) throws Exception;
12
13     /*按照查询条件分页查询科室信息记录*/
14     1 usage
15     public ArrayList<Department> queryDepartment(@Param("where") String where,@Param("startIndex") int startIndex,@Param("pageSize") int pageSize);
16
17     /*按照查询条件查询所有科室信息记录*/
18     2 usages
19     public ArrayList<Department> queryDepartmentList(@Param("where") String where) throws Exception;
20
21     /*按照查询条件的科室信息记录数*/
22     1 usage
23     public int queryDepartmentCount(@Param("where") String where) throws Exception;
24
25     /*根据主键查询某条科室信息记录*/
26     1 usage
27     public Department getDepartment(int departmentId) throws Exception;
28
29     /*更新科室信息记录*/
30     1 usage
31     public void updateDepartment(Department department) throws Exception;
32
33     /*删除科室信息记录*/
34     2 usages
35     public void deleteDepartment(int departmentId) throws Exception;
36 }

```

DoctorMapper:

```

1 package com.chengxusheji.mapper;
2
3 import java.util.ArrayList;
4 import org.apache.ibatis.annotations.Param;
5 import com.chengxusheji.po.Doctor;
6
7 public interface DoctorMapper {
8     /*添加医生信息*/
9     1 usage
10     public void addDoctor(Doctor doctor) throws Exception;
11
12     /*按照查询条件分页查询医生信息记录*/
13     1 usage
14     public ArrayList<Doctor> queryDoctor(@Param("where") String where,@Param("startIndex") int startIndex,@Param("pageSize") int pageSize) throw
15
16     /*按照查询条件查询所有医生信息记录*/
17     2 usages
18     public ArrayList<Doctor> queryDoctorList(@Param("where") String where) throws Exception;
19
20     /*按照查询条件的医生信息记录数*/
21     1 usage
22     public int queryDoctorCount(@Param("where") String where) throws Exception;
23
24     /*根据主键查询某条医生信息记录*/
25     1 usage
26     public Doctor getDoctor(String doctorNumber) throws Exception;
27
28     /*更新医生信息记录*/
29     1 usage
30     public void updateDoctor(Doctor doctor) throws Exception;
31
32     /*删除医生信息记录*/
33     2 usages
34     public void deleteDoctor(String doctorNumber) throws Exception;
35
36 }

```

LeavewordMapper:

```

1 package com.chengxusheji.mapper;
2
3 import java.util.ArrayList;
4 import org.apache.ibatis.annotations.Param;
5 import com.chengxusheji.po.Leaveword;
6
7 public interface LeavewordMapper {
8     /*添加留言信息*/
9     1 usage
10     public void addLeaveword(Leaveword leaveword) throws Exception;
11
12     /*按照查询条件分页查询留言记录*/
13     1 usage
14     public ArrayList<Leaveword> queryLeaveword(@Param("where") String where,@Param("startIndex") int startIndex,@Param("pageSize") int pageSize) th
15
16     /*按照查询条件查询所有留言记录*/
17     2 usages
18     public ArrayList<Leaveword> queryLeavewordList(@Param("where") String where) throws Exception;
19
20     /*按照查询条件的留言记录数*/
21     1 usage
22     public int queryLeavewordCount(@Param("where") String where) throws Exception;
23
24     /*根据主键查询某条留言记录*/
25     1 usage
26     public Leaveword getLeaveword(int leaveWordId) throws Exception;
27
28     /*更新留言记录*/
29     1 usage
30     public void updateLeaveword(Leaveword leaveword) throws Exception;
31
32     /*删除留言记录*/
33     2 usages
34     public void deleteLeaveword(int leaveWordId) throws Exception;
35
36 }

```

NewsMapper:

```

1 package com.chengxusheji.mapper;
2
3 import java.util.ArrayList;
4 import org.apache.ibatis.annotations.Param;
5 import com.chengxusheji.po.News;
6
7 public interface NewsMapper {
8     /*添加新闻信息*/
9     1 usage
10     public void addNews(News news) throws Exception;
11
12     /*按照查询条件分页查询新闻信息记录*/
13     1 usage
14     public ArrayList<News> queryNews(@Param("where") String where,@Param("startIndex") int startIndex,@Param("pageSize") int pageSize) throws Ex
15
16     /*按照查询条件查询所有新闻信息记录*/
17     2 usages
18     public ArrayList<News> queryNewsList(@Param("where") String where) throws Exception;
19
20     /*按照查询条件的新闻信息记录数*/
21     1 usage
22     public int queryNewsCount(@Param("where") String where) throws Exception;
23
24     /*根据主键查询某条新闻信息记录*/
25     1 usage
26     public News getNews(int newsId) throws Exception;
27
28     /*更新新闻信息记录*/
29     1 usage
30     public void updateNews(News news) throws Exception;
31
32     /*删除新闻信息记录*/
33     2 usages
34     public void deleteNews(int newsId) throws Exception;
35
36 }

```

OrderInfoMapper:

```

1 package com.chengxusheji.mapper;
2
3 import java.util.ArrayList;
4 import org.apache.ibatis.annotations.Param;
5 import com.chengxusheji.po.OrderInfo;
6
7 public interface OrderInfoMapper {
8     /*添加预约信息*/
9     1 usage
10     public void addOrderInfo(OrderInfo orderInfo) throws Exception;
11
12     /*按照查询条件分页查询预约信息记录*/
13     1 usage
14     public ArrayList<OrderInfo> queryOrderInfo(@Param("where") String where,@Param("startIndex") int startIndex,@Param("pageSize") int pageSize) th
15
16     /*按照查询条件查询所有预约信息记录*/
17     2 usages
18     public ArrayList<OrderInfo> queryOrderInfoList(@Param("where") String where) throws Exception;
19
20     /*按照查询条件的预约信息记录数*/
21     1 usage
22     public int queryOrderInfoCount(@Param("where") String where) throws Exception;
23
24     /*根据主键查询某条预约信息记录*/
25     1 usage
26     public OrderInfo getOrderInfo(int orderId) throws Exception;
27
28     /*更新预约信息记录*/
29     1 usage
30     public void updateOrderInfo(OrderInfo orderInfo) throws Exception;
31
32     /*删除预约信息记录*/
33     2 usages
34     public void deleteOrderInfo(int orderId) throws Exception;
35
36 }

```

PatientMapper:

```

1 package com.chengxusheji.mapper;
2
3 import java.util.ArrayList;
4 import org.apache.ibatis.annotations.Param;
5 import com.chengxusheji.po.Patient;
6
7 public interface PatientMapper {
8     /*添加病人信息*/
9     1 usage
10     public void addPatient(Patient patient) throws Exception;
11
12     /*按照查询条件分页查询病人信息记录*/
13     1 usage
14     public ArrayList<Patient> queryPatient(@Param("where") String where,@Param("startIndex") int startIndex,@Param("pageSize") int pageSize) throws Exception;
15
16     /*按照查询条件查询所有病人信息记录*/
17     2 usages
18     public ArrayList<Patient> queryPatientList(@Param("where") String where) throws Exception;
19
20     /*按照查询条件的病人信息记录数*/
21     1 usage
22     public int queryPatientCount(@Param("where") String where) throws Exception;
23
24     /*根据主键查询某条病人信息记录*/
25     1 usage
26     public Patient getPatient(int patientId) throws Exception;
27
28     /*更新病人信息记录*/
29     1 usage
30     public void updatePatient(Patient patient) throws Exception;
31
32     /*删除病人信息记录*/
33     2 usages
34     public void deletePatient(int patientId) throws Exception;
35 }

```

UserInfoMapper:

```

1 package com.chengxusheji.mapper;
2
3 import java.util.ArrayList;
4 import org.apache.ibatis.annotations.Param;
5 import com.chengxusheji.po.UserInfo;
6
7 public interface UserInfoMapper {
8     /*添加用户信息*/
9     1 usage
10     public void addUserInfo(UserInfo userInfo) throws Exception;
11
12     /*按照查询条件分页查询用户记录*/
13     1 usage
14     public ArrayList<UserInfo> queryUserInfo(@Param("where") String where,@Param("startIndex") int startIndex,@Param("pageSize") int pageSize) throws Exception;
15
16     /*按照查询条件查询所有用户记录*/
17     2 usages
18     public ArrayList<UserInfo> queryUserInfoList(@Param("where") String where) throws Exception;
19
20     /*按照查询条件的用户记录数*/
21     1 usage
22     public int queryUserInfoCount(@Param("where") String where) throws Exception;
23
24     /*根据主键查询某条用户记录*/
25     1 usage
26     public UserInfo getUserInfo(String user_name) throws Exception;
27
28     /*更新用户记录*/
29     1 usage
30     public void updateUserInfo(UserInfo userInfo) throws Exception;
31
32     /*删除用户记录*/
33     2 usages
34     public void deleteUserInfo(String user_name) throws Exception;
35 }

```

po 包:

Admin:

```

4      import org.hibernate.validator.constraints.NotEmpty;
5
6
7      28 usages
8      public class Admin {
9
10         2 usages
11         private Integer id;
12
13         /**
14          * 类型 0 管理员 1 医生 2 患者
15          */
16         2 usages
17         private Integer type;
18         /*管理员用户名*/
19         2 usages
20         @NotEmpty(message="用户名不能为空")
21         private String username;
22         /*登陆密码*/
23         2 usages
24         @NotEmpty(message="登陆密码不能为空")
25         private String password;
26
27         public Integer getId() { return id; }
28
29         public void setId(Integer id) { this.id = id; }
30
31         public Integer getType() { return type; }
32
33         public void setType(Integer type) { this.type = type; }
34
35         public String getUsername() { return username; }
36         public void setUsername(String username) { this.username = username; }
37         public String getPassword() { return password; }
38         public void setPassword(String password) { this.password = password; }
39     }
40
41
42
43
44
45
46
47
48
49
50
51
52

```

Department:

```

1 package com.chengxusheji.po;
2
3 import javax.validation.constraints.NotNull;
4 import org.hibernate.validator.constraints.NotEmpty;
5 import org.json.JSONException;
6 import org.json.JSONObject;
7
8 public class Department {
9     /*科室id*/
10    2 usages
11    private Integer departmentId;
12    public Integer getDepartmentId() { return departmentId; }
13
14    public void setDepartmentId(Integer departmentId) { this.departmentId = departmentId; }
15
16
17
18    /*科室名称*/
19    2 usages
20    @NotEmpty(message="科室名称不能为空")
21    private String departmentName;
22    public String getDepartmentName() { return departmentName; }
23    public void setDepartmentName(String departmentName) { this.departmentName = departmentName; }
24
25
26
27    /*科室介绍*/
28    2 usages
29    private String departmentDesc;
30    public String getDepartmentDesc() { return departmentDesc; }
31    public void setDepartmentDesc(String departmentDesc) { this.departmentDesc = departmentDesc; }
32
33
34
35
36
37    /*成立日期*/
38    2 usages
39    @NotEmpty(message="成立日期不能为空")
40    private String birthDate;
41    public String getBirthDate() { return birthDate; }
42    public void setBirthDate(String birthDate) { this.birthDate = birthDate; }
43
44
45
46
47    /*负责人*/
48    2 usages
49    private String chargeMan;
50    public String getChargeMan() { return chargeMan; }
51
52    public void setChargeMan(String chargeMan) { this.chargeMan = chargeMan; }
53
54
55
56    public JSONObject getJsonObject() throws JSONException {
57        JSONObject jsonDepartment=new JSONObject();
58        jsonDepartment.accumulate("departmentId", this.getDepartmentId());
59        jsonDepartment.accumulate("departmentName", this.getDepartmentName());
60        jsonDepartment.accumulate("departmentDesc", this.getDepartmentDesc());
61        jsonDepartment.accumulate("birthDate", this.getBirthDate().length()>19?this.getBirthDate().substring(0,19):this.getBirthDate());
62        jsonDepartment.accumulate("chargeMan", this.getChargeMan());
63        return jsonDepartment;
64    }
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99

```

Doctor:

```

1 package com.chengxusheji.po;
2
3 import javax.validation.constraints.NotNull;
4 import org.hibernate.validator.constraints.NotEmpty;
5 import org.json.JSONException;
6 import org.json.JSONObject;
7
8 public class Doctor {
9
10     2 usages
11     private Integer id;
12     2 usages
13     private Integer adminid;
14     /*医生工号*/
15     2 usages
16     @NotEmpty(message="医生工号不能为空")
17     private String doctorNumber;
18     public String getDoctorNumber() { return doctorNumber; }
19     public void setDoctorNumber(String doctorNumber) { this.doctorNumber = doctorNumber; }
20
21     /*登录密码*/
22     2 usages
23     private String password;
24     public String getPassword() { return password; }
25     public void setPassword(String password) { this.password = password; }
26
27     /*所在科室*/
28     2 usages
29     private Department departmentObj;
30     public Department getDepartmentObj() { return departmentObj; }
31     public void setDepartmentObj(Department departmentObj) { this.departmentObj = departmentObj; }
32
33     /*医生姓名*/
34     2 usages
35     @NotEmpty(message="医生姓名不能为空")
36     private String doctorName;
37     public String getDoctorName() { return doctorName; }
38     public void setDoctorName(String doctorName) { this.doctorName = doctorName; }
39
40     /*性别*/
41     2 usages
42     @NotEmpty(message="性别不能为空")
43     private String sex;
44     public String getSex() { return sex; }
45     public void setSex(String sex) { this.sex = sex; }
46
47     /*医生照片*/
48     2 usages
49     private String doctorPhoto;
50     4 usages
51     public String getDoctorPhoto() { return doctorPhoto; }
52     2 usages
53     public void setDoctorPhoto(String doctorPhoto) { this.doctorPhoto = doctorPhoto; }
54
55     /*出生日期*/
56     2 usages
57     @NotEmpty(message="出生日期不能为空")
58     private String birthDate;
59     public String getBirthDate() { return birthDate; }
60     public void setBirthDate(String birthDate) { this.birthDate = birthDate; }
61
62     /*医生职位*/
63     2 usages
64     @NotEmpty(message="医生职位不能为空")
65     private String position;
66     public String getPosition() { return position; }
67     public void setPosition(String position) { this.position = position; }
68
69     /*工作经验*/
70     2 usages
71     @NotEmpty(message="工作经验不能为空")
72     private String experience;
73     public String getExperience() { return experience; }
74     public void setExperience(String experience) { this.experience = experience; }
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98

```

```

99      /*联系方式*/
100      2 usages
101      private String contactWay;
102      public String getContactWay() { return contactWay; }
103      public void setContactWay(String contactWay) { this.contactWay = contactWay; }
104
105      /*擅长*/
106      2 usages
107      private String goodAt;
108      public String getGoodAt() { return goodAt; }
109      public void setGoodAt(String goodAt) { this.goodAt = goodAt; }
110
111      /*医生介绍*/
112      2 usages
113      @NotEmpty(message="医生介绍不能为空")
114      private String doctorDesc;
115      public String getDoctorDesc() { return doctorDesc; }
116      public void setDoctorDesc(String doctorDesc) { this.doctorDesc = doctorDesc; }
117
118      public Integer getId() { return id; }
119
120      public void setId(Integer id) { this.id = id; }
121
122      no usages
123      public Integer getAdminid() { return adminid; }
124
125      1 usage
126      public void setAdminid(Integer adminid) { this.adminid = adminid; }
127
128      public JSONObject getJsonObject() throws JSONException {
129          JSONObject jsonDoctor=new JSONObject();
130          jsonDoctor.accumulate("doctorNumber", this.getDoctorNumber());
131
132          jsonDoctor.accumulate("password", this.getPassword());
133          jsonDoctor.accumulate("departmentObj", this.getDepartmentObj().getDepartmentName());
134          jsonDoctor.accumulate("departmentObjPri", this.getDepartmentObj().getDepartmentId());
135          jsonDoctor.accumulate("doctorName", this.getDoctorName());
136          jsonDoctor.accumulate("sex", this.getSex());
137          jsonDoctor.accumulate("doctorPhoto", this.getDoctorPhoto());
138          jsonDoctor.accumulate("birthDate", this.getBirthDate().length()>19?this.getBirthDate().substring(0,19):this.getBirthDate());
139          jsonDoctor.accumulate("position", this.getPosition());
140          jsonDoctor.accumulate("experience", this.getExperience());
141          jsonDoctor.accumulate("contactWay", this.getContactWay());
142          jsonDoctor.accumulate("goodAt", this.getGoodAt());
143          jsonDoctor.accumulate("doctorDesc", this.getDoctorDesc());
144          return jsonDoctor;
145      }
146
147      }

```

Leaveword:

```

1 package com.chengxusheji.po;
2
3 import javax.validation.constraints.NotNull;
4 import org.hibernate.validator.constraints.NotEmpty;
5 import org.json.JSONException;
6 import org.json.JSONObject;
7
8 public class Leaveword {
9     /*留言id*/
10    2 usages
11    private Integer leaveWordId;
12    public Integer getLeaveWordId() { return leaveWordId; }
13
14    public void setLeaveWordId(Integer leaveWordId) { this.leaveWordId = leaveWordId; }
15
16
17
18    /*留言标题*/
19    2 usages
20    @NotEmpty(message="留言标题不能为空")
21    private String leaveTitle;
22    public String getLeaveTitle() { return leaveTitle; }
23
24    public void setLeaveTitle(String leaveTitle) { this.leaveTitle = leaveTitle; }
25
26
27
28    /*留言内容*/
29    2 usages
30    @NotEmpty(message="留言内容不能为空")
31    private String leaveContent;
32    public String getLeaveContent() { return leaveContent; }
33
34    public void setLeaveContent(String leaveContent) { this.leaveContent = leaveContent; }
35
36
37
38    /*留言人*/
39    2 usages
40    private UserInfo userObj;
41    public UserInfo getUserObj() { return userObj; }
42
43    public void setUserObj(UserInfo userObj) { this.userObj = userObj; }
44
45
46
47    /*留言时间*/
48    2 usages
49    private String leaveTime;
50    public String getLeaveTime() { return leaveTime; }
51
52    public void setLeaveTime(String leaveTime) { this.leaveTime = leaveTime; }
53
54
55
56    /*管理回复*/
57    2 usages
58    private String replyContent;
59    public String getReplyContent() { return replyContent; }
60
61    public void setReplyContent(String replyContent) { this.replyContent = replyContent; }
62
63
64
65    /*回复时间*/
66    2 usages
67    private String replyTime;
68    public String getReplyTime() { return replyTime; }
69
70    public void setReplyTime(String replyTime) { this.replyTime = replyTime; }
71
72
73
74    public JSONObject getJsonObject() throws JSONException {
75        JSONObject jsonLeaveword=new JSONObject();
76        jsonLeaveword.accumulate("LeaveWordId", this.getLeaveWordId());
77        jsonLeaveword.accumulate("LeaveTitle", this.getLeaveTitle());
78        jsonLeaveword.accumulate("leaveContent", this.getLeaveContent());
79        jsonLeaveword.accumulate("userObj", this.getUserObj().getName());
80        jsonLeaveword.accumulate("userObjPri", this.getUserObj().getUser_name());
81        jsonLeaveword.accumulate("leaveTime", this.getLeaveTime().length()>19?this.getLeaveTime().substring(0,19):this.getLeaveTime());
82        jsonLeaveword.accumulate("replyContent", this.getReplyContent());
83        jsonLeaveword.accumulate("replyTime", this.getReplyTime() != null?(this.getReplyTime().length()>19?this.getReplyTime().substring(0,19):this.getReplyTime()):null);
84        return jsonLeaveword;
85    }
86 }

```

News:

```

1 package com.chengxusheji.po;
2
3 import javax.validation.constraints.NotNull;
4 import org.hibernate.validator.constraints.NotEmpty;
5 import org.json.JSONException;
6 import org.json.JSONObject;
7
8 public class News {
9     /*新闻id*/
10    2 usages
11    private Integer newsId;
12    public Integer getNewsId() { return newsId; }
13    public void setNewsId(Integer newsId) { this.newsId = newsId; }
14
15    /*新闻标题*/
16    2 usages
17    @NotEmpty(message="新闻标题不能为空")
18    private String newsTitle;
19    public String getNewsTitle() { return newsTitle; }
20    public void setNewsTitle(String newsTitle) { this.newsTitle = newsTitle; }
21
22    /*新闻图片*/
23    2 usages
24    private String newsPhoto;
25    4 usages
26    public String getNewsPhoto() { return newsPhoto; }
27    2 usages
28    public void setNewsPhoto(String newsPhoto) { this.newsPhoto = newsPhoto; }
29
30    /*新闻内容*/
31    2 usages
32    @NotEmpty(message="新闻内容不能为空")
33    private String newsContent;
34    public String getNewsContent() { return newsContent; }
35    public void setNewsContent(String newsContent) { this.newsContent = newsContent; }
36
37    /*新闻日期*/
38    2 usages
39    @NotEmpty(message="新闻日期不能为空")
40    private String newsDate;
41    public String getNewsDate() { return newsDate; }
42    public void setNewsDate(String newsDate) { this.newsDate = newsDate; }
43
44    /*新闻来源*/
45    2 usages
46    private String newsFrom;
47    public String getNewsFrom() { return newsFrom; }
48    public void setNewsFrom(String newsFrom) { this.newsFrom = newsFrom; }
49
50    public JSONObject getJsonObject() throws JSONException {
51        JSONObject jsonNews=new JSONObject();
52        jsonNews.accumulate("newsId", this.getNewsId());
53        jsonNews.accumulate("newsTitle", this.getNewsTitle());
54        jsonNews.accumulate("newsPhoto", this.getNewsPhoto());
55        jsonNews.accumulate("newsContent", this.getNewsContent());
56        jsonNews.accumulate("newsDate", this.getNewsDate().length()>19?this.getNewsDate().substring(0,19):this.getNewsDate());
57        jsonNews.accumulate("newsFrom", this.getNewsFrom());
58        return jsonNews;
59    }
60 }

```

OrderInfo:

```

1 package com.chengxusheji.po;
2
3 import javax.validation.constraints.NotNull;
4 import org.hibernate.validator.constraints.NotEmpty;
5 import org.json.JSONException;
6 import org.json.JSONObject;
7
8 public class OrderInfo {
9     /*预约id*/
10    2 usages
11    private Integer orderId;
12    public Integer getOrderId() { return orderId; }
13
14    public void setOrderId(Integer orderId) { this.orderId = orderId; }
15
16
17
18    /*预约用户*/
19    2 usages
20    private UserInfo userObj;
21    public UserInfo getUserObj() { return userObj; }
22
23    public void setUserObj(UserInfo userObj) { this.userObj = userObj; }
24
25
26
27    /*预约医生*/
28    2 usages
29    private Doctor doctorObj;
30    public Doctor getDoctorObj() { return doctorObj; }
31
32    public void setDoctorObj(Doctor doctorObj) { this.doctorObj = doctorObj; }
33
34
35
36    /*预约日期*/
37    2 usages
38    @NotEmpty(message="预约日期不能为空")
39    private String orderDate;
40    public String getOrderDate() { return orderDate; }
41
42    public void setOrderDate(String orderDate) { this.orderDate = orderDate; }
43
44
45
46    /*时段*/
47    2 usages
48    @NotEmpty(message="时段不能为空")
49    private String timeInterval;
50    public String getTimeInterval() { return timeInterval; }
51
52    public void setTimeInterval(String timeInterval) { this.timeInterval = timeInterval; }
53
54
55
56    /*联系电话*/
57    2 usages
58    @NotEmpty(message="联系电话不能为空")
59    private String telephone;
60    public String getTelephone() { return telephone; }
61
62    public void setTelephone(String telephone) { this.telephone = telephone; }
63
64
65
66    /*下单时间*/
67    2 usages
68    @NotEmpty(message="下单时间不能为空")
69    private String orderTime;
70    public String getOrderTime() { return orderTime; }
71
72    public void setOrderTime(String orderTime) { this.orderTime = orderTime; }
73
74
75
76    /*处理状态*/
77    2 usages
78    @NotEmpty(message="处理状态不能为空")
79    private String checkState;
80    public String getCheckState() { return checkState; }
81
82    public void setCheckState(String checkState) { this.checkState = checkState; }
83
84
85
86    /*医生回复*/
87    2 usages
88    @NotEmpty(message="医生回复不能为空")
89    private String replyContent;
90    public String getReplyContent() { return replyContent; }
91
92    public void setReplyContent(String replyContent) { this.replyContent = replyContent; }
93
94
95
96    public JSONObject getJsonObject() throws JSONException {
97        JSONObject jsonOrderInfo=new JSONObject();
98        jsonOrderInfo.accumulate("orderId", this.getOrderId());

```

```

99         jsonOrderInfo.accumulate("userObj", this.getUserObj().getName());
100         jsonOrderInfo.accumulate("userObjPri", this.getUserObj().getUser_name());
101         jsonOrderInfo.accumulate("doctorObj", this.getDoctorObj().getDoctorName());
102         jsonOrderInfo.accumulate("doctorObjPri", this.getDoctorObj().getDoctorNumber());
103         jsonOrderInfo.accumulate("orderDate", this.getOrderDate().length()>19?this.getOrderDate().substring(0,19):this.getOrderDate());
104         jsonOrderInfo.accumulate("timeInterval", this.getTimeInterval());
105         jsonOrderInfo.accumulate("telephone", this.getTelephone());
106         jsonOrderInfo.accumulate("orderTime", this.getOrderTime().length()>19?this.getOrderTime().substring(0,19):this.getOrderTime());
107         jsonOrderInfo.accumulate("checkState", this.getCheckState());
108         jsonOrderInfo.accumulate("replyContent", this.getReplyContent());
109         return jsonOrderInfo;
110     }}

```

Patient:

```

Patient.java x
1  package com.chengxusheji.po;
2
3  import javax.validation.constraints.NotNull;
4  import org.hibernate.validator.constraints.NotEmpty;
5  import org.json.JSONException;
6  import org.json.JSONObject;
7
8  public class Patient {
9      /*病人id*/
10     2 usages
11     private Integer patientId;
12     public Integer getPatientId() { return patientId; }
13     public void setPatientId(Integer patientId) { this.patientId = patientId; }
14
15     /*医生*/
16     2 usages
17     private Doctor doctorObj;
18     public Doctor getDoctorObj() { return doctorObj; }
19     public void setDoctorObj(Doctor doctorObj) { this.doctorObj = doctorObj; }
20
21     /*病人姓名*/
22     2 usages
23     @NotEmpty(message="病人姓名不能为空")
24     private String patientName;
25     public String getPatientName() { return patientName; }
26     public void setPatientName(String patientName) { this.patientName = patientName; }
27
28     /*性别*/
29     2 usages
30     @NotEmpty(message="性别不能为空")
31     private String sex;
32     public String getSex() { return sex; }
33     public void setSex(String sex) { this.sex = sex; }
34
35     /*身份证号*/
36     2 usages
37     @NotEmpty(message="身份证号不能为空")
38     private String cardNumber;
39

```

```

50 public String getCardNumber() { return cardNumber; }
53 public void setCardNumber(String cardNumber) { this.cardNumber = cardNumber; }
56
57 /*联系电话*/
58 2 usages
59 @NotEmpty(message="联系电话不能为空")
60 private String telephone;
63 public String getTelephone() { return telephone; }
66 public void setTelephone(String telephone) { this.telephone = telephone; }
69
70 /*病人病例*/
71 2 usages
72 @NotEmpty(message="病人病例不能为空")
73 private String illnessCase;
76 public String getIllnessCase() { return illnessCase; }
79 public void setIllnessCase(String illnessCase) { this.illnessCase = illnessCase; }
82
83 /*登记时间*/
84 2 usages
85 @NotEmpty(message="登记时间不能为空")
86 private String addTime;
89 public String getAddTime() { return addTime; }
92 public void setAddTime(String addTime) { this.addTime = addTime; }
95
96 public JSONObject getJsonObject() throws JSONException {
97     JSONObject jsonPatient=new JSONObject();
98     jsonPatient.accumulate("patientId", this.getPatientId());
99     jsonPatient.accumulate("doctorObj", this.getDoctorObj().getDoctorName());
100     jsonPatient.accumulate("doctorObjPri", this.getDoctorObj().getDoctorNumber());
101     jsonPatient.accumulate("patientName", this.getPatientName());
102     jsonPatient.accumulate("sex", this.getSex());
103     jsonPatient.accumulate("cardNumber", this.getCardNumber());
104     jsonPatient.accumulate("telephone", this.getTelephone());
105     jsonPatient.accumulate("illnessCase", this.getIllnessCase());
106     jsonPatient.accumulate("addTime", this.getAddTime().length()>19?this.getAddTime().substring(0,19):this.getAddTime());
107     return jsonPatient;
108 }

```

UserInfo:

```

1 package com.chengxusheji.po;
2
3 import org.hibernate.validator.constraints.NotEmpty;
4 import org.json.JSONException;
5 import org.json.JSONObject;
6
7 public class UserInfo {
8
9     2 usages
10    private Integer id;
11    2 usages
12    private Integer adminid;
13    /*用户名*/
14    2 usages
15    @NotEmpty(message="用户名不能为空")
16    private String user_name;
17    public String getUser_name() { return user_name; }
18    public void setUser_name(String user_name) { this.user_name = user_name; }
19
20    /*登录密码*/
21    2 usages
22    @NotEmpty(message="登录密码不能为空")
23    private String password;
24    public String getPassword() { return password; }
25    public void setPassword(String password) { this.password = password; }
26
27    /*姓名*/
28    2 usages
29    @NotEmpty(message="姓名不能为空")
30    private String name;
31    public String getName() { return name; }
32    public void setName(String name) { this.name = name; }
33
34    /*性别*/
35    2 usages
36    @NotEmpty(message="性别不能为空")
37    private String gender;
38    public String getGender() { return gender; }
39    public void setGender(String gender) { this.gender = gender; }
40
41    /*出生日期*/
42    2 usages
43    @NotEmpty(message="出生日期不能为空")
44    private String birthDate;
45    public String getBirthDate() { return birthDate; }
46    public void setBirthDate(String birthDate) { this.birthDate = birthDate; }
47
48    /*用户照片*/
49    2 usages
50    private String userPhoto;
51    public String getUserPhoto() { return userPhoto; }
52    public void setUserPhoto(String userPhoto) { this.userPhoto = userPhoto; }
53
54    /*联系电话*/
55    2 usages
56    @NotEmpty(message="联系电话不能为空")
57    private String telephone;
58    public String getTelephone() { return telephone; }
59    public void setTelephone(String telephone) { this.telephone = telephone; }
60
61    /*邮箱*/
62    2 usages
63    @NotEmpty(message="邮箱不能为空")
64    private String email;
65    public String getEmail() { return email; }
66    public void setEmail(String email) { this.email = email; }
67
68    /*家庭地址*/
69    2 usages
70    private String address;
71    public String getAddress() { return address; }
72    public void setAddress(String address) { this.address = address; }
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98

```

```

99      /*注册时间*/
100      2 usages
101      private String regTime;
102      public String getRegTime() { return regTime; }
103      public void setRegTime(String regTime) { this.regTime = regTime; }
104      public Integer getId() { return id; }
105
106      public void setId(Integer id) { this.id = id; }
107
108      1 usage
109      public Integer getAdminid() { return adminid; }
110
111      1 usage
112      public void setAdminid(Integer adminid) { this.adminid = adminid; }
113      public JSONObject getJsonObject() throws JSONException {
114          JSONObject jsonUserInfo=new JSONObject();
115          jsonUserInfo.accumulate("id", this.getId());
116          jsonUserInfo.accumulate("adminid", this.getAdminid());
117          jsonUserInfo.accumulate("user_name", this.getUser_name());
118          jsonUserInfo.accumulate("password", this.getPassword());
119          jsonUserInfo.accumulate("name", this.getName());
120          jsonUserInfo.accumulate("gender", this.getGender());
121          jsonUserInfo.accumulate("birthDate", this.getBirthDate().length()>19?this.getBirthDate().substring(0,19):this.getBirthDate());
122          jsonUserInfo.accumulate("userPhoto", this.getUserPhoto());
123          jsonUserInfo.accumulate("telephone", this.getTelephone());
124          jsonUserInfo.accumulate("email", this.getEmail());
125          jsonUserInfo.accumulate("address", this.getAddress());
126          jsonUserInfo.accumulate("regTime", this.getRegTime().length()>19?this.getRegTime().substring(0,19):this.getRegTime());
127          return jsonUserInfo;
128      }
129  }

```

Service 包:

AdminService:

```

AdminService.java
1  package com.chengxusheji.service;
2
3  import com.chengxusheji.po.UserInfo;
4  import javax.annotation.Resource;
5
6
7  import org.springframework.stereotype.Service;
8
9  import com.chengxusheji.mapper.AdminMapper;
10 import com.chengxusheji.po.Admin;
11
12  6 usages
13  @Service
14  public class AdminService {
15      4 usages
16      @Resource AdminMapper adminMapper;
17
18      /*保存业务逻辑错误信息字段*/
19      5 usages
20      private String errMessage;
21      1 usage
22      public String getErrMessage() { return this.errMessage; }
23
24      /*验证用户登录*/
25      2 usages
26      public Admin checkLogin(Admin admin) throws Exception {
27          Admin db_admin = (Admin) adminMapper.findAdminByUserName(admin.getUsername());
28          if(db_admin == null) {
29              this.errMessage = " 账号不存在 ";
30              System.out.print(this.errMessage);
31              return null;
32          } else if( !db_admin.getPassword().equals(admin.getPassword())) {
33              this.errMessage = " 密码不正确! ";
34              System.out.print(this.errMessage);
35              return null;
36          }
37      }
38  }

```

```

33         return db_admin;
34     }
35
36     /*添加用户记录*/
37     3 usages
38     public void addAdmin(Admin admin) throws Exception {
39         adminMapper.addAdmin(admin);
40     }
41
42     /*修改用户登录密码*/
43     1 usage
44     public void changePassword(String username, String newPassword) throws Exception {
45         Admin admin = new Admin();
46         admin.setUsername(username);
47         admin.setPassword(newPassword);
48         adminMapper.changePassword(admin);
49     }
50
51     /*根据用户名获取管理员对象*/
52     6 usages
53     public Admin findAdminByUserName(String username) throws Exception {
54         Admin db_admin = null;
55         db_admin = adminMapper.findAdminByUserName(username);
56         return db_admin;
57     }

```

DepartmentSeivice:

```

DepartmentService.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.Department;
7
8  import com.chengxusheji.mapper.DepartmentMapper;
9  4 usages
10 @Service
11 public class DepartmentService {
12
13     9 usages
14     @Resource DepartmentMapper departmentMapper;
15     /*每页显示记录数目*/
16     6 usages
17     private int rows = 10;;
18     public int getRows() { return rows; }
19     public void setRows(int rows) { this.rows = rows; }
20
21     /*保存查询后总的页数*/
22     4 usages
23     private int totalPage;
24     public void setTotalPage(int totalPage) { this.totalPage = totalPage; }
25     public int getTotalPage() { return totalPage; }
26
27     /*保存查询到的总记录数*/
28     5 usages
29     private int recordNumber;
30     public void setRecordNumber(int recordNumber) { this.recordNumber = recordNumber; }
31     public int getRecordNumber() { return recordNumber; }
32
33     /*添加科室信息记录*/
34     1 usage
35     public void addDepartment(Department department) throws Exception {
36         departmentMapper.addDepartment(department);
37     }
38
39

```

```

44
45      /*按照查询条件分页查询科室信息记录*/
46      2 usages
47      @ public ArrayList<Department> queryDepartment(String departmentName,String birthDate,int currentPage) throws Exception {
48          String where = "where 1=1";
49          if(!departmentName.equals("")) where = where + " and t_department.departmentName like '%" + departmentName + "%'";
50          if(!birthDate.equals("")) where = where + " and t_department.birthDate like '%" + birthDate + "%'";
51          int startIndex = (currentPage-1) * this.rows;
52          return departmentMapper.queryDepartment(where, startIndex, this.rows);
53      }
54
55      /*按照查询条件查询所有记录*/
56      1 usage
57      @ public ArrayList<Department> queryDepartment(String departmentName,String birthDate) throws Exception {
58          String where = "where 1=1";
59          if(!departmentName.equals("")) where = where + " and t_department.departmentName like '%" + departmentName + "%'";
60          if(!birthDate.equals("")) where = where + " and t_department.birthDate like '%" + birthDate + "%'";
61          return departmentMapper.queryDepartmentList(where);
62      }
63
64      /*查询所有科室信息记录*/
65      4 usages
66      @ public ArrayList<Department> queryAllDepartment() throws Exception {
67          return departmentMapper.queryDepartmentList( where: "where 1=1");
68      }
69
70      /*当前查询条件下计算总的页数和记录数*/
71      @ public void queryTotalPageAndRecordNumber(String departmentName,String birthDate) throws Exception {
72          String where = "where 1=1";
73          if(!departmentName.equals("")) where = where + " and t_department.departmentName like '%" + departmentName + "%'";
74          if(!birthDate.equals("")) where = where + " and t_department.birthDate like '%" + birthDate + "%'";
75          recordNumber = departmentMapper.queryDepartmentCount(where);
76          int mod = recordNumber % this.rows;
77          totalPage = recordNumber / this.rows;
78          if(mod != 0) totalPage++;
79
80      }
81
82      /*根据主键获取科室信息记录*/
83      2 usages
84      @ public Department getDepartment(int departmentId) throws Exception {
85          Department department = departmentMapper.getDepartment(departmentId);
86          return department;
87      }
88
89      /*更新科室信息记录*/
90      1 usage
91      @ public void updateDepartment(Department department) throws Exception {
92          departmentMapper.updateDepartment(department);
93      }
94
95      /*删除一条科室信息记录*/
96      1 usage
97      @ public void deleteDepartment (int departmentId) throws Exception {
98          departmentMapper.deleteDepartment(departmentId);
99      }
100
101      /*删除多条科室信息信息*/
102      1 usage
103      @ public int deleteDepartments (String departmentIds) throws Exception {
104          String _departmentIds[] = departmentIds.split( regex: "%");
105          for(String _departmentId: _departmentIds) {
106              departmentMapper.deleteDepartment(Integer.parseInt(_departmentId));
107          }
108          return _departmentIds.length;
109      }
110
111      }
112
113

```

DoctorService:

```

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.Department;
7 import com.chengxusheji.po.Doctor;
8
9 import com.chengxusheji.mapper.DoctorMapper;
10
11 @Service
12 public class DoctorService {
13
14     @Resource DoctorMapper doctorMapper;
15     /*每页显示记录数目*/
16     private int rows = 10;;
17     public int getRows() { return rows; }
18     public void setRows(int rows) { this.rows = rows; }
19
20     /*保存查询后总的页数*/
21     private int totalPage;
22     public void setTotalPage(int totalPage) { this.totalPage = totalPage; }
23     public int getTotalPage() { return totalPage; }
24
25     /*保存查询到的总记录数*/
26     private int recordNumber;
27     public void setRecordNumber(int recordNumber) { this.recordNumber = recordNumber; }
28     public int getRecordNumber() { return recordNumber; }
29
30     /*添加医生信息记录*/
31     public void addDoctor(Doctor doctor) throws Exception {
32         doctorMapper.addDoctor(doctor);
33     }
34
35     /*按照查询条件分页查询医生信息记录*/
36     public ArrayList<Doctor> queryDoctor(String doctorNumber, Department departmentObj, String doctorName, String birthDate, int currentPage) throws Exception {
37         String where = "where 1=1";
38         if(!doctorNumber.equals("")) where = where + " and t_doctor.doctorNumber like '%" + doctorNumber + "%'";
39         if(null != departmentObj && departmentObj.getDepartmentId() != null && departmentObj.getDepartmentId() != 0) where += " and t_doctor.departmentId=" + departmentObj.getDepartmentId();
40         if(!doctorName.equals("")) where = where + " and t_doctor.doctorName like '%" + doctorName + "%'";
41         if(!birthDate.equals("")) where = where + " and t_doctor.birthDate like '%" + birthDate + "%'";
42         int startIndex = (currentPage-1) * this.rows;
43         return doctorMapper.queryDoctor(where, startIndex, this.rows);
44     }
45
46     /*按照查询条件查询所有记录*/
47     public ArrayList<Doctor> queryDoctor(String doctorNumber, Department departmentObj, String doctorName, String birthDate) throws Exception {
48         String where = "where 1=1";
49         if(!doctorNumber.equals("")) where = where + " and t_doctor.doctorNumber like '%" + doctorNumber + "%'";
50         if(null != departmentObj && departmentObj.getDepartmentId() != null && departmentObj.getDepartmentId() != 0) where += " and t_doctor.departmentId=" + departmentObj.getDepartmentId();
51         if(!doctorName.equals("")) where = where + " and t_doctor.doctorName like '%" + doctorName + "%'";
52         if(!birthDate.equals("")) where = where + " and t_doctor.birthDate like '%" + birthDate + "%'";
53         return doctorMapper.queryDoctorList(where);
54     }
55
56     /*查询所有医生信息记录*/
57     public ArrayList<Doctor> queryAllDoctor() throws Exception {
58         return doctorMapper.queryDoctorList("where 1=1");
59     }
60
61     /*当前查询条件下计算总的页数和记录数*/
62     public void queryTotalPageAndRecordNumber(String doctorNumber, Department departmentObj, String doctorName, String birthDate) throws Exception {
63         String where = "where 1=1";
64         if(!doctorNumber.equals("")) where = where + " and t_doctor.doctorNumber like '%" + doctorNumber + "%'";
65         if(null != departmentObj && departmentObj.getDepartmentId() != null && departmentObj.getDepartmentId() != 0) where += " and t_doctor.departmentId=" + departmentObj.getDepartmentId();
66         if(!doctorName.equals("")) where = where + " and t_doctor.doctorName like '%" + doctorName + "%'";
67         if(!birthDate.equals("")) where = where + " and t_doctor.birthDate like '%" + birthDate + "%'";
68         return doctorMapper.queryTotalPageAndRecordNumber(where);
69     }
70 }

```

```

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*/
86     3 usages
87     public Doctor getDoctor(String doctorNumber) throws Exception {
88         Doctor doctor = doctorMapper.getDoctor(doctorNumber);
89         return doctor;
90     }
91
92     /*更新医生信息记录*/
93     1 usage
94     public void updateDoctor(Doctor doctor) throws Exception {
95         doctorMapper.updateDoctor(doctor);
96     }
97
98     /*删除一条医生信息记录*/
99     1 usage
100     public void deleteDoctor (String doctorNumber) throws Exception {
101         doctorMapper.deleteDoctor(doctorNumber);
102     }
103
104     /*删除多条医生信息信息*/
105     1 usage
106     @ public int deleteDoctors (String doctorNumbers) throws Exception {
107         String _doctorNumbers[] = doctorNumbers.split( regex: "," );
108         for(String _doctorNumber: _doctorNumbers) {
109             doctorMapper.deleteDoctor(_doctorNumber);
110         }
111         return _doctorNumbers.length;
112     }

```

LeavewordService:

```

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.Leaveword;
8
9 import com.chengxusheji.mapper.LeavewordMapper;
10
11 @Service
12 public class LeavewordService {
13
14     @Resource LeavewordMapper leavewordMapper;
15     /*每页显示记录数目*/
16     private int rows = 10;;
17     public int getRows() { return rows; }
18     public void setRows(int rows) { this.rows = rows; }
19
20     /*保存查询后总的页数*/
21     private int totalPage;
22     public void setTotalPage(int totalPage) { this.totalPage = totalPage; }
23     public int getTotalPage() { return totalPage; }
24
25     /*保存查询到的总记录数*/
26     private int recordNumber;
27     public void setRecordNumber(int recordNumber) { this.recordNumber = recordNumber; }
28     public int getRecordNumber() { return recordNumber; }
29
30     /*添加留言记录*/
31     public void addLeaveword(Leaveword leaveword) throws Exception {
32         leavewordMapper.addLeaveword(leaveword);
33     }
34 }

```

```

35
36 /*按照查询条件分页查询留言记录*/
37 @ public ArrayList<Leaveword> queryLeaveword(String leaveTitle,UserInfo userObj,String leaveTime,int currentPage) throws Exception {
38     String where = "where 1=1";
39     if(!leaveTitle.equals("")) where = where + " and t_leaveword.leaveTitle like '%" + leaveTitle + "%'";
40     if(!userObj.getUser_name().equals("")) where += " and t_leaveword.userObj='" + userObj.getUser_name() + "'";
41     if(!leaveTime.equals("")) where = where + " and t_leaveword.leaveTime like '%" + leaveTime + "%'";
42     int startIndex = (currentPage-1) * this.rows;
43     return leavewordMapper.queryLeaveword(where, startIndex, this.rows);
44 }
45
46 /*按照查询条件查询所有记录*/
47 @ public ArrayList<Leaveword> queryLeavewordList(String leaveTitle,UserInfo userObj,String leaveTime) throws Exception {
48     String where = "where 1=1";
49     if(!leaveTitle.equals("")) where = where + " and t_leaveword.leaveTitle like '%" + leaveTitle + "%'";
50     if(!userObj.getUser_name().equals("")) where += " and t_leaveword.userObj='" + userObj.getUser_name() + "'";
51     if(!leaveTime.equals("")) where = where + " and t_leaveword.leaveTime like '%" + leaveTime + "%'";
52     return leavewordMapper.queryLeavewordList(where);
53 }
54
55 /*查询所有留言记录*/
56 public ArrayList<Leaveword> queryAllLeaveword() throws Exception {
57     return leavewordMapper.queryLeavewordList( "where 1=1");
58 }
59
60 /*当前查询条件下计算总的页数和记录数*/
61 @ public void queryTotalPageAndRecordNumber(String leaveTitle,UserInfo userObj,String leaveTime) throws Exception {
62     String where = "where 1=1";
63     if(!leaveTitle.equals("")) where = where + " and t_leaveword.leaveTitle like '%" + leaveTitle + "%'";
64     if(!userObj.getUser_name().equals("")) where += " and t_leaveword.userObj='" + userObj.getUser_name() + "'";
65     if(!leaveTime.equals("")) where = where + " and t_leaveword.leaveTime like '%" + leaveTime + "%'";
66     recordNumber = leavewordMapper.queryLeavewordCount(where);
67     int mod = recordNumber % this.rows;
68 }

```

```

78         totalPages = recordNumber / this.rows;
79         if(mod != 0) totalPages++;
80     }
81
82     /*根据主键获取留言记录*/
83     2 usages
84     public Leaveword getLeaveword(int leaveWordId) throws Exception {
85         Leaveword leaveword = leavewordMapper.getLeaveword(leaveWordId);
86         return leaveword;
87     }
88
89     /*更新留言记录*/
90     1 usage
91     public void updateLeaveword(Leaveword leaveword) throws Exception {
92         leavewordMapper.updateLeaveword(leaveword);
93     }
94
95     /*删除一条留言记录*/
96     1 usage
97     public void deleteLeaveword (int leaveWordId) throws Exception {
98         leavewordMapper.deleteLeaveword(leaveWordId);
99     }
100
101     /*删除多条留言信息*/
102     1 usage
103     @ public int deleteLeavewords (String leaveWordIds) throws Exception {
104         String _leaveWordIds[] = leaveWordIds.split( regex: " ");
105         for(String _leaveWordId: _leaveWordIds) {
106             leavewordMapper.deleteLeaveword(Integer.parseInt(_leaveWordId));
107         }
108         return _leaveWordIds.length;
109     }
110

```

NewsService:

```

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.News;
7
8 import com.chengxusheji.mapper.NewsMapper;
9
10 @Service
11 public class NewsService {
12
13     9 usages
14     @Resource NewsMapper newsMapper;
15     /*每页显示记录数目*/
16     6 usages
17     private int rows = 10;;
18     public int getRows() { return rows; }
19     public void setRows(int rows) { this.rows = rows; }
20
21     /*保存查询后总的页数*/
22     4 usages
23     private int totalPage;
24     public void setTotalPage(int totalPage) { this.totalPage = totalPage; }
25     public int getTotalPage() { return totalPage; }
26
27     /*保存查询到的总记录数*/
28     5 usages
29     private int recordNumber;
30     public void setRecordNumber(int recordNumber) { this.recordNumber = recordNumber; }
31     public int getRecordNumber() { return recordNumber; }
32
33     /*添加新闻信息记录*/
34     1 usage
35     public void addNews(News news) throws Exception {
36         newsMapper.addNews(news);
37     }
38
39     /*按照查询条件分页查询新闻信息记录*/
40     2 usages
41     @ public ArrayList<News> queryNews(String newsTitle,String newsDate,int currentPage) throws Exception {
42         String where = "where 1=1";
43         if(!newsTitle.equals("")) where = where + " and t_news.newsTitle like '%" + newsTitle + "%'";
44         if(!newsDate.equals("")) where = where + " and t_news.newsDate like '%" + newsDate + "%'";
45         int startIndex = (currentPage-1) * this.rows;
46         return newsMapper.queryNews(where, startIndex, this.rows);
47     }
48
49     /*按照查询条件查询所有记录*/
50     1 usage
51     @ public ArrayList<News> queryNews(String newsTitle,String newsDate) throws Exception {
52         String where = "where 1=1";
53         if(!newsTitle.equals("")) where = where + " and t_news.newsTitle like '%" + newsTitle + "%'";
54         if(!newsDate.equals("")) where = where + " and t_news.newsDate like '%" + newsDate + "%'";
55         return newsMapper.queryNewsList(where);
56     }
57
58     /*查询所有新闻信息记录*/
59     1 usage
60     public ArrayList<News> queryAllNews() throws Exception {
61         return newsMapper.queryNewsList( where: "where 1=1");
62     }
63
64     /*当前查询条件下计算总的页数和记录数*/
65     @ public void queryTotalPageAndRecordNumber(String newsTitle,String newsDate) throws Exception {
66         String where = "where 1=1";
67         if(!newsTitle.equals("")) where = where + " and t_news.newsTitle like '%" + newsTitle + "%'";
68         if(!newsDate.equals("")) where = where + " and t_news.newsDate like '%" + newsDate + "%'";
69         recordNumber = newsMapper.queryNewsCount(where);
70         int mod = recordNumber % this.rows;
71         totalPage = recordNumber / this.rows;
72         if(mod != 0) totalPage++;
73     }
74
75
76

```

```

77
78     /*根据主键获取新闻信息记录*/
79     2 usages
80     public News getNews(int newsId) throws Exception {
81         News news = newsMapper.getNews(newsId);
82         return news;
83     }
84
85     /*更新新闻信息记录*/
86     1 usage
87     public void updateNews(News news) throws Exception {
88         newsMapper.updateNews(news);
89     }
90
91     /*删除一条新闻信息记录*/
92     1 usage
93     public void deleteNews (int newsId) throws Exception {
94         newsMapper.deleteNews(newsId);
95     }
96
97     /*删除多条新闻信息信息*/
98     1 usage
99     public int deleteNews (String newsIds) throws Exception {
100         String _newsIds[] = newsIds.split( regex: "," );
101         for(String _newsId: _newsIds) {
102             newsMapper.deleteNews(Integer.parseInt(_newsId));
103         }
104         return _newsIds.length;
105     }
106 }
107
108
109

```

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;
11    2 usages
12    @Service
13    public class OrderInfoService {
14
15        9 usages
16        @Resource OrderInfoMapper orderInfoMapper;
17        /*每页显示记录数目*/
18        6 usages
19        private int rows = 10;;
20        public int getRows() { return rows; }
21        public void setRows(int rows) { this.rows = rows; }
22
23        /*保存查询后总的页数*/
24        4 usages
25        private int totalPage;
26        public void setTotalPage(int totalPage) { this.totalPage = totalPage; }
27        public int getTotalPage() { return totalPage; }
28
29        /*保存查询到的总记录数*/
30        5 usages
31        private int recordNumber;
32        public void setRecordNumber(int recordNumber) { this.recordNumber = recordNumber; }
33        public int getRecordNumber() { return recordNumber; }
34
35
36

```

```

41
42      /*添加预约信息记录*/
43      1 usage
44      public void addOrderInfo(OrderInfo orderInfo) throws Exception {
45          orderInfoMapper.addOrderInfo(orderInfo);
46      }
47
48      /*按照查询条件分页查询预约信息记录*/
49      2 usages
50      public ArrayList<OrderInfo> queryOrderInfo(UserInfo userObj, Doctor doctorObj, String orderDate, String timeInterval, String telephone, String checkState) throws Exception {
51          String where = "where 1=1";
52          if(null != userObj && userObj.getUser_name() != null && !userObj.getUser_name().equals("")) where += " and t_orderInfo.userObj='" + userObj.getUser_name() + "'";
53          if(null != doctorObj && doctorObj.getDoctorNumber() != null && !doctorObj.getDoctorNumber().equals("")) where += " and t_orderInfo.doctorObj='" + doctorObj.getDoctorNumber() + "'";
54          if(!orderDate.equals("")) where = where + " and t_orderInfo.orderDate like '%" + orderDate + "%'";
55          if(!timeInterval.equals("")) where = where + " and t_orderInfo.timeInterval like '%" + timeInterval + "%'";
56          if(!telephone.equals("")) where = where + " and t_orderInfo.telephone like '%" + telephone + "%'";
57          if(!checkState.equals("")) where = where + " and t_orderInfo.checkState like '%" + checkState + "%'";
58          if(!orderTime.equals("")) where = where + " and t_orderInfo.orderTime like '%" + orderTime + "%'";
59          int startIndex = (currentPage-1) * this.rows;
60          return orderInfoMapper.queryOrderInfo(where, startIndex, this.rows);
61      }
62
63      /*按照查询条件查询所有记录*/
64      1 usage
65      public ArrayList<OrderInfo> queryOrderInfoList(UserInfo userObj, Doctor doctorObj, String orderDate, String timeInterval, String telephone, String checkState) throws Exception {
66          String where = "where 1=1";
67          if(null != userObj && userObj.getUser_name() != null && !userObj.getUser_name().equals("")) where += " and t_orderInfo.userObj='" + userObj.getUser_name() + "'";
68          if(null != doctorObj && doctorObj.getDoctorNumber() != null && !doctorObj.getDoctorNumber().equals("")) where += " and t_orderInfo.doctorObj='" + doctorObj.getDoctorNumber() + "'";
69          if(!orderDate.equals("")) where = where + " and t_orderInfo.orderDate like '%" + orderDate + "%'";
70          if(!timeInterval.equals("")) where = where + " and t_orderInfo.timeInterval like '%" + timeInterval + "%'";
71          if(!telephone.equals("")) where = where + " and t_orderInfo.telephone like '%" + telephone + "%'";
72          if(!checkState.equals("")) where = where + " and t_orderInfo.checkState like '%" + checkState + "%'";
73          if(!orderTime.equals("")) where = where + " and t_orderInfo.orderTime like '%" + orderTime + "%'";
74          return orderInfoMapper.queryOrderInfoList(where);
75      }

```

```

74
75      /*查询所有预约信息记录*/
76      1 usage
77      public ArrayList<OrderInfo> queryAllOrderInfo() throws Exception {
78          return orderInfoMapper.queryOrderInfoList("where 1=1");
79      }
80
81      /*当前查询条件下计算总页数和记录数*/
82      public void queryTotalPageAndRecordNumber(UserInfo userObj, Doctor doctorObj, String orderDate, String timeInterval, String telephone, String checkState) throws Exception {
83          String where = "where 1=1";
84          if(null != userObj && userObj.getUser_name() != null && !userObj.getUser_name().equals("")) where += " and t_orderInfo.userObj='" + userObj.getUser_name() + "'";
85          if(null != doctorObj && doctorObj.getDoctorNumber() != null && !doctorObj.getDoctorNumber().equals("")) where += " and t_orderInfo.doctorObj='" + doctorObj.getDoctorNumber() + "'";
86          if(!orderDate.equals("")) where = where + " and t_orderInfo.orderDate like '%" + orderDate + "%'";
87          if(!timeInterval.equals("")) where = where + " and t_orderInfo.timeInterval like '%" + timeInterval + "%'";
88          if(!telephone.equals("")) where = where + " and t_orderInfo.telephone like '%" + telephone + "%'";
89          if(!checkState.equals("")) where = where + " and t_orderInfo.checkState like '%" + checkState + "%'";
90          if(!orderTime.equals("")) where = where + " and t_orderInfo.orderTime like '%" + orderTime + "%'";
91          recordNumber = orderInfoMapper.queryOrderInfoCount(where);
92          int mod = recordNumber % this.rows;
93          totalPage = recordNumber / this.rows;
94          if(mod != 0) totalPage++;
95      }
96
97      /*根据主键获取预约信息记录*/
98      2 usages
99      public OrderInfo getOrderInfo(int orderId) throws Exception {
100          OrderInfo orderInfo = orderInfoMapper.getOrderInfo(orderId);
101          return orderInfo;
102      }

```

```

101      /*更新预约信息记录*/
102      1 usage
103      public void updateOrderInfo(OrderInfo orderInfo) throws Exception {
104          orderInfoMapper.updateOrderInfo(orderInfo);
105      }
106
107      /*删除一条预约信息记录*/
108      1 usage
109      public void deleteOrderInfo (int orderId) throws Exception {
110          orderInfoMapper.deleteOrderInfo(orderId);
111      }
112
113      /*删除多条预约信息信息*/
114      1 usage
115      @ public int deleteOrderInfos (String orderIds) throws Exception {
116          String _orderIds[] = orderIds.split( regex "," );
117          for(String _orderId: _orderIds) {
118              orderInfoMapper.deleteOrderInfo(Integer.parseInt(_orderId));
119          }
120          return _orderIds.length;
121      }
122
123      }
124
125      }

```

PatientService:

```

PatientService.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.Doctor;
7      import com.chengxusheji.po.Patient;
8
9      import com.chengxusheji.mapper.PatientMapper;
10     2 usages
11     @Service
12     public class PatientService {
13
14         9 usages
15         @Resource PatientMapper patientMapper;
16         /*每页显示记录数目*/
17         6 usages
18         private int rows = 10;;
19         public int getRows() { return rows; }
20         public void setRows(int rows) { this.rows = rows; }
21
22
23         /*保存查询后总的页数*/
24         4 usages
25         private int totalPage;
26         public void setTotalPage(int totalPage) { this.totalPage = totalPage; }
27         public int getTotalPage() { return totalPage; }
28
29
30         /*保存查询到的总记录数*/
31         5 usages
32         private int recordNumber;
33         public void setRecordNumber(int recordNumber) { this.recordNumber = recordNumber; }
34         public int getRecordNumber() { return recordNumber; }
35
36
37         /*添加病人信息记录*/
38         1 usage
39         public void addPatient(Patient patient) throws Exception {
40             patientMapper.addPatient(patient);
41         }
42     }

```

```

46  /*按照查询条件分页查询病人信息记录*/
47  2 usages
48  public ArrayList<Patient> queryPatient(Doctor doctorObj,String patientName,String cardNumber,String telephone,String addTime,int currentPage) {
49      String where = "where 1=1";
50      if(null != doctorObj && doctorObj.getDoctorNumber() != null && !doctorObj.getDoctorNumber().equals("")) where += " and t_patient.doctorId=" + doctorObj.getDoctorNumber();
51      if(!patientName.equals("")) where = where + " and t_patient.patientName like '%" + patientName + "%'";
52      if(!cardNumber.equals("")) where = where + " and t_patient.cardNumber like '%" + cardNumber + "%'";
53      if(!telephone.equals("")) where = where + " and t_patient.telephone like '%" + telephone + "%'";
54      if(!addTime.equals("")) where = where + " and t_patient.addTime like '%" + addTime + "%'";
55      int startIndex = (currentPage-1) * this.rows;
56      return patientMapper.queryPatient(where, startIndex, this.rows);
57  }
58
59  /*按照查询条件查询所有记录*/
60  1 usage
61  public ArrayList<Patient> queryPatient(Doctor doctorObj,String patientName,String cardNumber,String telephone,String addTime) throws Exception {
62      String where = "where 1=1";
63      if(null != doctorObj && doctorObj.getDoctorNumber() != null && !doctorObj.getDoctorNumber().equals("")) where += " and t_patient.doctorId=" + doctorObj.getDoctorNumber();
64      if(!patientName.equals("")) where = where + " and t_patient.patientName like '%" + patientName + "%'";
65      if(!cardNumber.equals("")) where = where + " and t_patient.cardNumber like '%" + cardNumber + "%'";
66      if(!telephone.equals("")) where = where + " and t_patient.telephone like '%" + telephone + "%'";
67      if(!addTime.equals("")) where = where + " and t_patient.addTime like '%" + addTime + "%'";
68      return patientMapper.queryPatientList(where);
69  }
70
71  /*查询所有病人信息记录*/
72  1 usage
73  public ArrayList<Patient> queryAllPatient() throws Exception {
74      return patientMapper.queryPatientList( where: "where 1=1");
75  }
76
77  /*当前查询条件下计算总的页数和记录数*/
78  1 usage
79  public void queryTotalPageAndRecordNumber(Doctor doctorObj,String patientName,String cardNumber,String telephone,String addTime) throws Exception {
80      String where = "where 1=1";
81      if(null != doctorObj && doctorObj.getDoctorNumber() != null && !doctorObj.getDoctorNumber().equals("")) where += " and t_patient.doctorId=" + doctorObj.getDoctorNumber();
82      if(!patientName.equals("")) where = where + " and t_patient.patientName like '%" + patientName + "%'";
83      if(!cardNumber.equals("")) where = where + " and t_patient.cardNumber like '%" + cardNumber + "%'";
84      if(!telephone.equals("")) where = where + " and t_patient.telephone like '%" + telephone + "%'";
85      if(!addTime.equals("")) where = where + " and t_patient.addTime like '%" + addTime + "%'";
86      recordNumber = patientMapper.queryPatientCount(where);
87      int mod = recordNumber % this.rows;
88      totalPage = recordNumber / this.rows;
89      if(mod != 0) totalPage++;
90  }
91
92  /*根据主键获取病人信息记录*/
93  2 usages
94  public Patient getPatient(int patientId) throws Exception {
95      Patient patient = patientMapper.getPatient(patientId);
96      return patient;
97  }
98
99  /*更新病人信息记录*/
100  1 usage
101  public void updatePatient(Patient patient) throws Exception {
102      patientMapper.updatePatient(patient);
103  }
104
105  /*删除一条病人信息记录*/
106  1 usage
107  public void deletePatient (int patientId) throws Exception {
108      patientMapper.deletePatient(patientId);
109  }
110
111  /*删除多条病人信息信息*/
112  1 usage
113  public int deletePatients (String patientIds) throws Exception {
114      String _patientIds[] = patientIds.split( regex: "，");
115      for(String _patientId: _patientIds) {
116          patientMapper.deletePatient(Integer.parseInt(_patientId));
117      }
118      return _patientIds.length;
119  }

```

UserInfoService:

```

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
8 import com.chengxusheji.mapper.UserInfoMapper;
9
10 @Service
11 public class UserInfoService {
12
13     9 usages
14     @Resource UserInfoMapper userInfoMapper;
15     /*每页显示记录数目*/
16     6 usages
17     private int rows = 10;;
18     public int getRows() { return rows; }
19     public void setRows(int rows) { this.rows = rows; }
20
21     /*保存查询后总的页数*/
22     4 usages
23     private int totalPage;
24     public void setTotalPage(int totalPage) { this.totalPage = totalPage; }
25     public int getTotalPage() { return totalPage; }
26
27     /*保存查询到的总记录数*/
28     5 usages
29     private int recordNumber;
30     public void setRecordNumber(int recordNumber) { this.recordNumber = recordNumber; }
31     public int getRecordNumber() { return recordNumber; }
32
33     /*添加用户记录*/
34     1 usage
35     public void addUserInfo(UserInfo userInfo) throws Exception {
36         userInfoMapper.addUserInfo(userInfo);
37     }

```

```

38
39     /*按照查询条件分页查询用户记录*/
40     2 usages
41     public ArrayList<UserInfo> queryUserInfo(String user_name,String name,String birthDate,String telephone,int currentPage) throws Exception {
42         String where = "where 1=1";
43         if(!user_name.equals("")) where = where + " and t_userInfo.user_name like '%" + user_name + "%'";
44         if(!name.equals("")) where = where + " and t_userInfo.name like '%" + name + "%'";
45         if(!birthDate.equals("")) where = where + " and t_userInfo.birthDate like '%" + birthDate + "%'";
46         if(!telephone.equals("")) where = where + " and t_userInfo.telephone like '%" + telephone + "%'";
47         int startIndex = (currentPage-1) * this.rows;
48         return userInfoMapper.queryUserInfo(where, startIndex, this.rows);
49     }
50
51     /*按照查询条件查询所有记录*/
52     1 usage
53     public ArrayList<UserInfo> queryUserInfo(String user_name,String name,String birthDate,String telephone) throws Exception {
54         String where = "where 1=1";
55         if(!user_name.equals("")) where = where + " and t_userInfo.user_name like '%" + user_name + "%'";
56         if(!name.equals("")) where = where + " and t_userInfo.name like '%" + name + "%'";
57         if(!birthDate.equals("")) where = where + " and t_userInfo.birthDate like '%" + birthDate + "%'";
58         if(!telephone.equals("")) where = where + " and t_userInfo.telephone like '%" + telephone + "%'";
59         return userInfoMapper.queryUserInfolist(where);
60     }
61
62     /*查询所有用户记录*/
63     7 usages
64     public ArrayList<UserInfo> queryAllUserInfo() throws Exception {
65         return userInfoMapper.queryUserInfolist( where: "where 1=1");
66     }
67
68     /*当前查询条件下计算总的页数和记录数*/
69     public void queryTotalPageAndRecordNumber(String user_name,String name,String birthDate,String telephone) throws Exception {
70         String where = "where 1=1";
71         if(!user_name.equals("")) where = where + " and t_userInfo.user_name like '%" + user_name + "%'";
72         if(!name.equals("")) where = where + " and t_userInfo.name like '%" + name + "%'";
73         if(!birthDate.equals("")) where = where + " and t_userInfo.birthDate like '%" + birthDate + "%'";
74         if(!telephone.equals("")) where = where + " and t_userInfo.telephone like '%" + telephone + "%'";

```

```

78         recordNumber = userInfoMapper.queryUserInfoCount(where);
79         int mod = recordNumber % this.rows;
80         totalPage = recordNumber / this.rows;
81         if(mod != 0) totalPage++;
82     }
83
84     /*根据主键获取用户记录*/
85     3 usages
86     public UserInfo getUserInfo(String user_name) throws Exception {
87         UserInfo userInfo = userInfoMapper.getUserInfo(user_name);
88         return userInfo;
89     }
90
91     /*更新用户记录*/
92     1 usage
93     public void updateUserInfo(UserInfo userInfo) throws Exception {
94         userInfoMapper.updateUserInfo(userInfo);
95     }
96
97     /*删除一条用户记录*/
98     1 usage
99     public void deleteUserInfo (String user_name) throws Exception {
100         userInfoMapper.deleteUserInfo(user_name);
101     }
102
103     /*删除多条用户信息*/
104     1 usage
105     @ public int deleteUserInfos (String user_names) throws Exception {
106         String _user_names[] = user_names.split( regex: "/" );
107         for(String _user_name: _user_names) {
108             userInfoMapper.deleteUserInfo(_user_name);
109         }
110         return _user_names.length;
111     }
112 }

```

util 包:

ExportExcelUtil:

```

1 package com.chengxusheji.utils;
2
3 import java.io.BufferedInputStream;
4 import java.io.FileInputStream;
5 import java.io.IOException;
6 import java.io.OutputStream;
7 import java.util.Collection;
8 import java.util.Iterator;
9
10 import org.apache.poi.hssf.usermodel.HSSFCell;
11 import org.apache.poi.hssf.usermodel.HSSFCellStyle;
12 import org.apache.poi.hssf.usermodel.HSSFClientAnchor;
13 import org.apache.poi.hssf.usermodel.HSSFComment;
14 import org.apache.poi.hssf.usermodel.HSSFFont;
15 import org.apache.poi.hssf.usermodel.HSSFPatriarch;
16 import org.apache.poi.hssf.usermodel.HSSFRichTextString;
17 import org.apache.poi.hssf.usermodel.HSSFRow;
18 import org.apache.poi.hssf.usermodel.HSSFSheet;
19 import org.apache.poi.hssf.usermodel.HSSFWorkbook;
20 import org.apache.poi.hssf.util.HSSFColor;
21
22 21 usages
23 public class ExportExcelUtil {
24
25     public void exportExcel(Collection<String[]> dataset, OutputStream out) {
26         exportExcel( rootPath: "", title: "测试POI导出EXCEL文档", headers: null, dataset, out, pattern: "yyyy-MM-dd");
27     }
28
29     public void exportExcel(String rootPath,String title, String[] headers,
30         Collection<String[]> dataset, OutputStream out) {
31         exportExcel(rootPath,title, headers, dataset, out, pattern: "yyyy-MM-dd");
32     }
33
34     public void exportExcel(String[] headers, Collection<String[]> dataset,
35         OutputStream out, String pattern) {
36         exportExcel( rootPath: "", title: "测试POI导出EXCEL文档", headers, dataset, out, pattern);
37     }
38 }

```

```

38 /**
39  * 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
40  *
41  * @param title
42  *      表格标题名
43  * @param headers
44  *      表格属性列名数组
45  * @param dataset
46  *      需要显示的数据集合
47  * @param out
48  *      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
49  * @param pattern
50  *      如果有时间数据，设定输出格式。默认为"yyy-MM-dd"
51  */
52 @unchecked
53 @ public void exportExcel(String rootPath,String title, String[] headers,
54     Collection<String[]> dataset, OutputStream out, String pattern) {
55     // 声明一个工作簿
56     HSSFWorkbook workbook = new HSSFWorkbook();
57     // 生成一个表格
58     HSSFSheet sheet = workbook.createSheet(title);
59     // 设置表格默认列宽度为15个字节
60     sheet.setDefaultColumnWidth((short) 15);
61     // 生成一个样式
62     HSSFCellStyle style = workbook.createCellStyle();
63     // 设置这些样式
64     style.setFillForegroundColor(HSSFColor.SKY_BLUE.index);
65     style.setFillPattern(HSSFCellStyle.SOLID_FOREGROUND);
66     style.setBorderBottom(HSSFCellStyle.BORDER_THIN);
67     style.setBorderLeft(HSSFCellStyle.BORDER_THIN);
68     style.setBorderRight(HSSFCellStyle.BORDER_THIN);
69     style.setBorderTop(HSSFCellStyle.BORDER_THIN);
70     style.setAlignment(HSSFCellStyle.ALIGN_CENTER);
71     // 生成一个字体
72     HSSFFont font = workbook.createFont();

```

```

73     font.setColor(HSSFColor.VIOLET.index);
74     font.setFontHeightInPoints((short) 12);
75     font.setBoldweight(HSSFFont.BOLDWEIGHT_BOLD);
76     // 把字体应用到当前的样式
77     style.setFont(font);
78     // 生成并设置另一个样式
79     HSSFCellStyle style2 = workbook.createCellStyle();
80     style2.setFillForegroundColor(HSSFColor.LIGHT_YELLOW.index);
81     style2.setFillPattern(HSSFCellStyle.SOLID_FOREGROUND);
82     style2.setBorderBottom(HSSFCellStyle.BORDER_THIN);
83     style2.setBorderLeft(HSSFCellStyle.BORDER_THIN);
84     style2.setBorderRight(HSSFCellStyle.BORDER_THIN);
85     style2.setBorderTop(HSSFCellStyle.BORDER_THIN);
86     style2.setAlignment(HSSFCellStyle.ALIGN_CENTER);
87     style2.setVerticalAlignment(HSSFCellStyle.VERTICAL_CENTER);
88     // 生成另一个字体
89     HSSFFont font2 = workbook.createFont();
90     font2.setBoldweight(HSSFFont.BOLDWEIGHT_NORMAL);
91     // 把字体应用到当前的样式
92     style2.setFont(font2);
93
94     // 声明一个画图的顶级管理器
95     HSSFPatriarch patriarch = sheet.createDrawingPatriarch();
96     // 定义注释的大小和位置, 详见文档
97     HSSFComment comment = patriarch.createComment(new HSSFClientAnchor( dx1: 0,
98         dy1: 0, dx2: 0, dy2: 0, (short) 4, row1: 2, (short) 6, row2: 5));
99     // 设置注释内容
100    comment.setString(new HSSFRichTextString("可以在POI中添加注释!"));
101    // 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。
102    comment.setAuthor("lenc");
103
104    // 产生表格标题行
105    HSSFRow row = sheet.createRow( rownum: 0);
106    for (short i = 0; i < headers.length; i++) {
107        HSSFCell cell = row.createCell(i);
108        cell.setCellStyle(style);
109        HSSFRichTextString text = new HSSFRichTextString(headers[i]);

```

```

110        cell.setCellValue(text);
111    }
112
113    // 遍历集合数据, 产生数据行
114    Iterator<String[]> it = dataset.iterator();
115    int index = 0;
116    while (it.hasNext()) {
117        index++;
118        row = sheet.createRow(index);
119        String[] t = (String[]) it.next();
120        for (short i = 0; i < t.length; i++) {
121            HSSFCell cell = row.createCell(i);
122            cell.setCellStyle(style2);
123            // 判断值的类型后进行强制类型转换
124            String textValue = t[i];
125
126            if(textValue.startsWith("upload/"))
127            {
128                // 有图片时, 设置行高为50px;
129                row.setHeightInPoints(50);
130                // 设置图片所在列宽度为80px, 注意这里单位的一个换算
131                sheet.setColumnWidth(i, (short) (35.7 * 80)); //
132                //sheet.autoSizeColumn(i);
133                BufferedInputStream bis;
134                byte[] buf = null;
135                try {
136                    bis = new BufferedInputStream(
137                        new FileInputStream( name: rootPath + textValue));
138                    buf = new byte[bis.available()];
139                    while ((bis.read(buf)) != -1) {}
140
141                    HSSFClientAnchor anchor = new HSSFClientAnchor( dx1: 0, dy1: 0, dx2: 1023,
142                        dy2: 255, (short) i, index, (short) i, index);
143                    anchor.setAnchorType(2);
144                    patriarch.createPicture(anchor,
145                        workbook.addPicture( buf,

```

```

146         HSSFWorkbook.PICTURE_TYPE_JPEG));
147     } catch (Exception e) {
148         e.printStackTrace();
149     }
150
151     } else {
152         HSSFRichTextString richString = new HSSFRichTextString(
153             textValue);
154         HSSFFont font3 = workbook.createFont();
155         font3.setColor(HSSFColor.BLUE.index);
156         richString.applyFont(font3);
157         cell.setCellValue(richString);
158     }
159
160     /*
161     * else if (value instanceof byte[]) { // 有图片时，设置行高为60px;
162     * 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
163     * sheet.setColumnWidth(1, (short) (35.7 * 80)); //
164     * sheet.autoSizeColumn(1); byte[] bsValue = (byte[]) value;
165     * HSSFClientAnchor anchor = new HSSFClientAnchor(0, 0, 1023,
166     * 255, (short) 6, index, (short) 6, index);
167     * anchor.setAnchorType(2); patriarch.createPicture(anchor,
168     * workbook.addPicture( bsValue,
169     * HSSFWorkbook.PICTURE_TYPE_JPEG)); } else{ //其它数据类型都当作字符串简单处理
170     * textValue = value.toString(); }
171     */
172
173     }
174 }
175
176 try {
177     workbook.write(out);
178 } catch (IOException e) {
179     // TODO Auto-generated catch block
180     e.printStackTrace();
181 }
182
183 }
184

```

UserException:

```

1 package com.chengxusheji.utils;
2
3 public class UserException extends RuntimeException {
4
5     /**
6      *
7      */
8     private static final long serialVersionUID = 1L;
9
10    public UserException() {
11        super();
12        // TODO Auto-generated constructor stub
13    }
14
15    public UserException(String message, Throwable cause) {
16        super(message, cause);
17        // TODO Auto-generated constructor stub
18    }
19
20    public UserException(String message) {
21        super(message);
22        // TODO Auto-generated constructor stub
23    }
24
25    public UserException(Throwable cause) {
26        super(cause);
27        // TODO Auto-generated constructor stub
28    }
29
30

```

Web.xml 配置文件:

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app version="2.5"
3     xmlns="http://java.sun.com/xml/ns/javaee"
4     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5     xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
6     http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
7
8     <!-- 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 -->
9     <context-param>
10         <param-name>contextConfigLocation</param-name>
11         <param-value>/WEB-INF/classes/spring/applicationContext-*.xml</param-value>
12     </context-param>
13
14
15     <!-- 对Spring容器进行实例化 -->
16     <listener>
17         <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
18     </listener>
19
20
21     <!-- springmvc前端控制器，rest配置 -->
22     <servlet>
23         <servlet-name>springmvc</servlet-name>
24         <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
25         <!-- 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，默认加载WEB-INF/classes/spring/springmvc.xml -->
26         <init-param>
27             <param-name>contextConfigLocation</param-name>
28             <param-value>classpath:spring/springmvc.xml</param-value>
29         </init-param>
30     </servlet>
31     <servlet-mapping>
32         <servlet-name>springmvc</servlet-name>
33         <url-pattern>/</url-pattern>
34     </servlet-mapping>
35
36

```

```

37 <filter>
38   <filter-name>CharacterFilter</filter-name>
39   <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
40   <init-param>
41     <param-name>encoding</param-name>
42     <param-value>UTF-8</param-value>
43   </init-param>
44 </filter>
45 <filter-mapping>
46   <filter-name>CharacterFilter</filter-name>
47   <url-pattern>/*</url-pattern>
48 </filter-mapping>
49
50
51 <welcome-file-list>
52   <welcome-file>index.jsp</welcome-file>
53 </welcome-file-list>
54 </web-app>
55

```

3. JavaWeb 项目运行

首先运行前数据库各个表的情况如下：

localhost	
information_schema	
hospital	192.0 KiB
admin	16.0 KiB
t_department	16.0 KiB
t_doctor	16.0 KiB
t_leaveword	32.0 KiB
t_news	16.0 KiB
t_orderinfo	48.0 KiB
t_patient	32.0 KiB
t_userinfo	16.0 KiB

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	dfsdfsdfs	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:

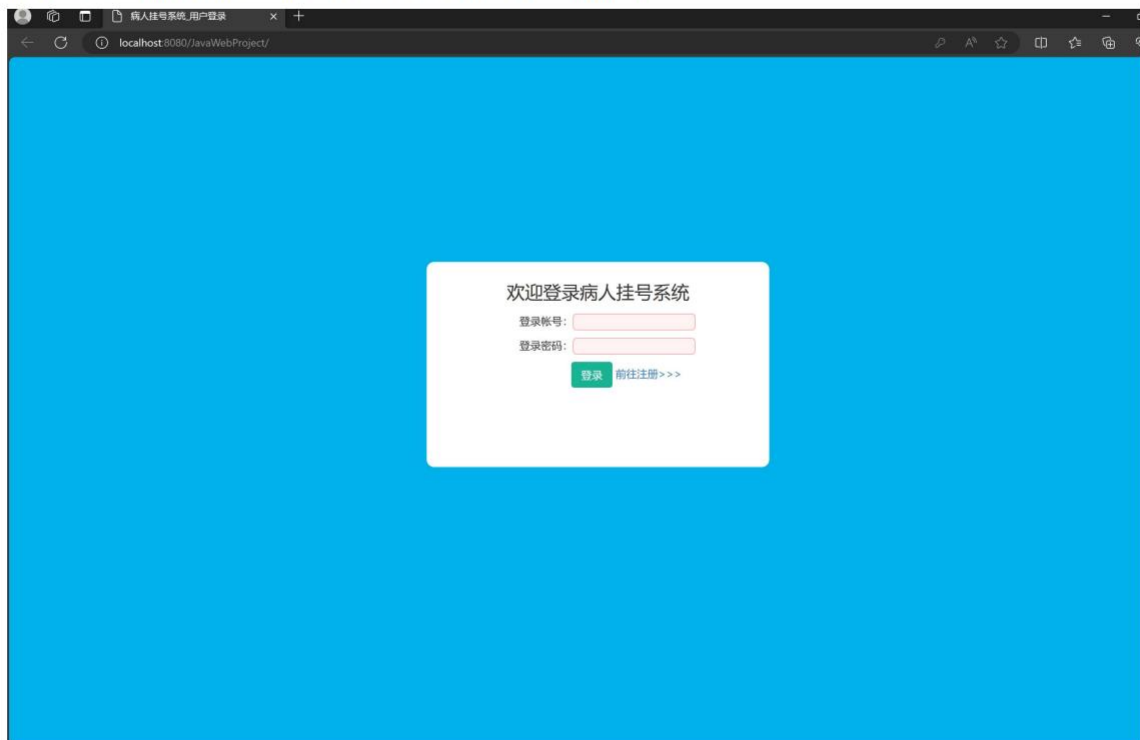
patientId	doctorObj	patientName	sex	cardNumber	telephone	illnessCase	addTime
-----------	-----------	-------------	-----	------------	-----------	-------------	---------

userinfo 表:

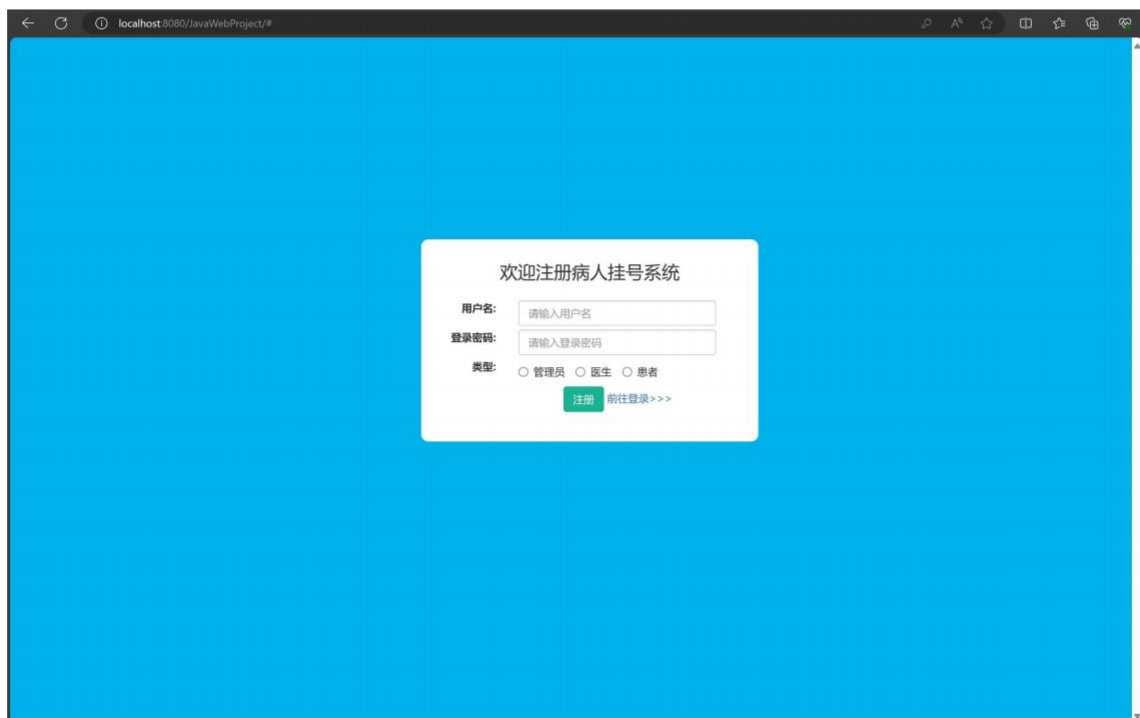
id	name	gender	birthDate	userPhoto	telephone	email	address	regTime	adminid	user_name	password
1	e	男	2023-12-30	upload/NoImage.jpg	131	1313	123	2023-12-30 21:18:41	5	e	e

正常运行各功能展示:

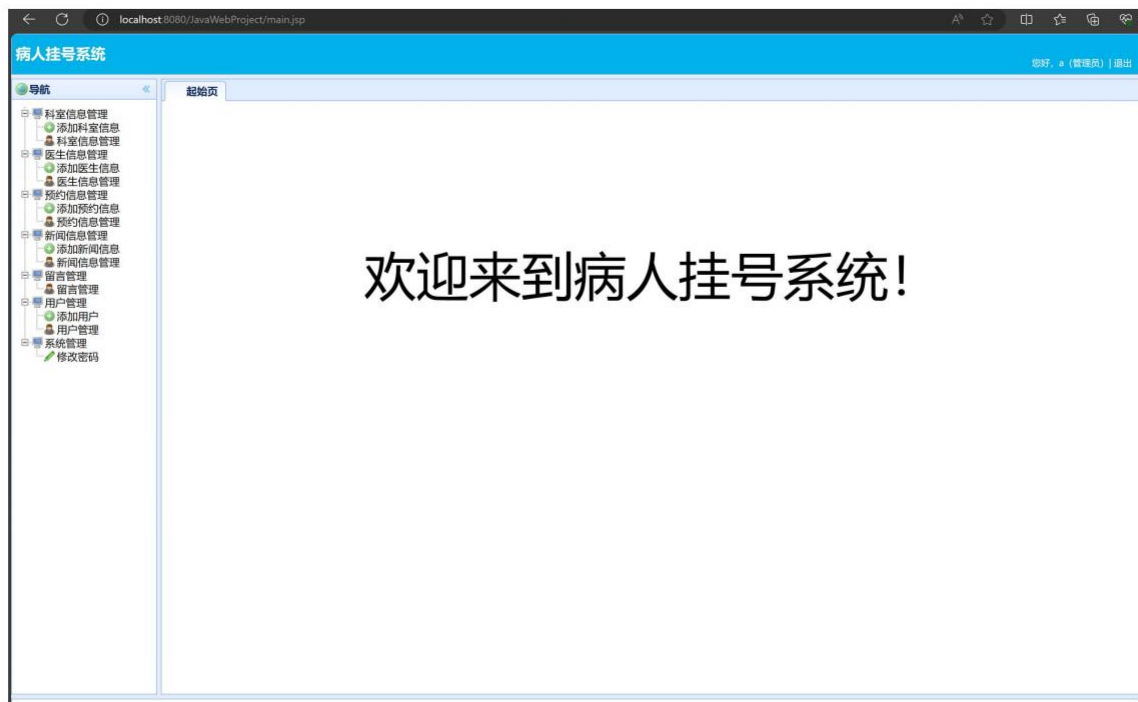
首先进入登陆页面，这时可以直接登陆或者进行注册



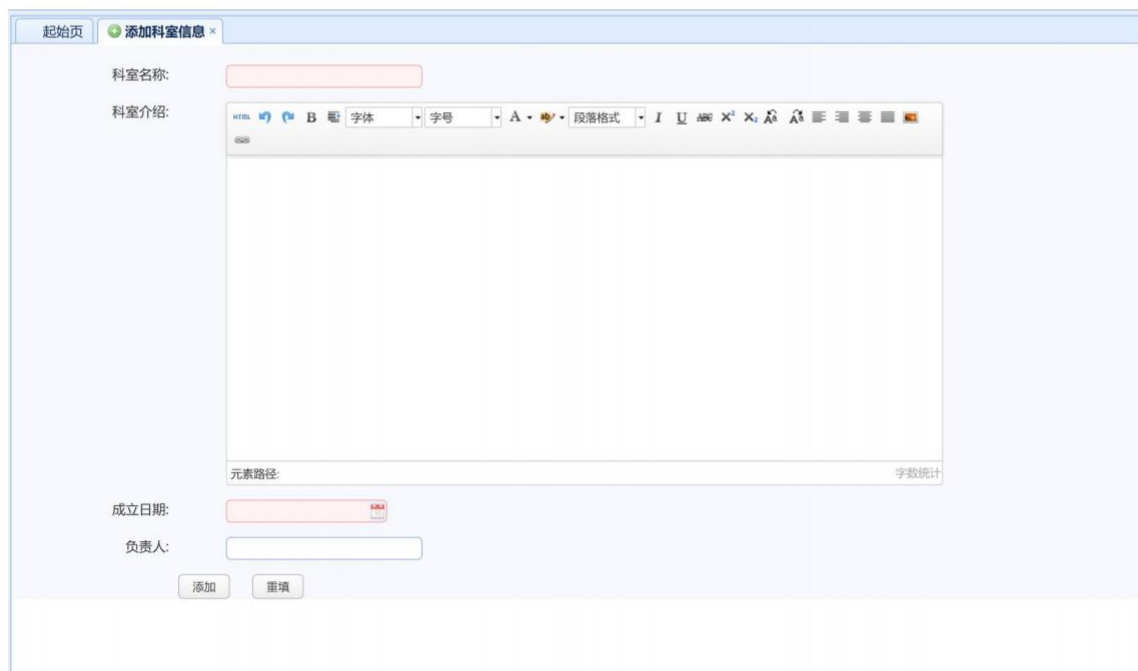
点击不同的单选按钮会出现不同的注册信息填写要求（管理员、医生、患者），这里选择管理员身份进行后续流程：



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:

HTML 富文本 字体 字号 段落格式 加粗 下划线 背景色 链接 取消链接 有序列表 无序列表 代码 全屏 打印 帮助

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: 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	暂无照片	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 ×

预约信息首理 ×

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

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:

H1 H2 B 超 字体 字号 A 段落格式 I U ABC X² X₂ A⁺ A⁻ 背景色 全屏 打印 帮助

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	新闻		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	暂无照片	131	1313	2023-12-30 21:18:41	

起始页 用户管理 × 科室信息管理 × 修改密码 ×

以前的密码:

输入新密码:

再输入新密码:

该系统其他页面以及细节展示：

欢迎登录病人挂号系统

登录帐号:

登录密码:

登录失败!

 密码不正确!



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：

科室id	科室名称	成立日期	负责人
2	内科	2023-12-30	内科
3	外科	2023-12-30	外科
4	皮肤科	2023-12-30	皮肤科
5	dddd	2023-12-30	aaaaa

A	B	C	D	E
1	科室id	科室名称	成立日期	负责人
2	2	内科	2023-12-30	内科
3	3	外科	2023-12-30	外科
4	4	皮肤科	2023-12-30	皮肤科
5	5	dddd	2023-12-30	aaaaa
6				
7				
8				
9				

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。