

安徽科技学院



《Java 框架技术》 课程报告

题 目: 基于 ssm 的健康信息管理系统

姓 名: 方子航

学 院: 信息与网络工程学院

专 业: 计算机科学与技术

班 级: 2021 级 2 班

学 号: 2701210205

授课教师 赵靖

2024 年 6 月 9 日

安徽科技学院教务处制

课程论文（设计）基本信息


学号 2701210205 姓名 方子航 班级计科 212

论文课题：基于 ssm 的健康信息管理系统

论文摘要：该系统使用的是MVC架构模式，能对系统的功能进行分离，对系统分成五个层次。其中持久对象层是由若干持久化类（实体类）组成。数据访问层由若干DAO接口和MyBatis映射文件组成。业务逻辑层由若干Service接口和实现类组成。Web表现层主要包括Spring MVC中的Controller类和JSP（html）页面。其中Controller是应用程序中处理用户交互的部分是系统的控制层，用户通过系统对数据的访问通过控制层对数据的控制之后通过模型处理再返回数据给控制层之后反馈给用户就能获取信息。

项目网址：<https://github.com/Limerence1314/ssmfangzihang>

评分细节表

项目	项目比例	分数（百分制）
平时成绩	20%	
实验	30%	
论文报告	50%	
评语		
总评	教师签名： 	

目 录

第 1 章 课程目的与要求	1
1.1 课程目的	1
1.2 实验环境	1
1.3 预备知识	1
1.4 课程要求	1
第 2 章 课程设计内容	2
2.1 需求分析	2
2.1.1 用户需求分析	2
2.1.2 功能需求分析	2
2.2 总体设计	3
2.3 详细设计	4
2.3.1 数据库设计	4
2.3.2 程序结构设计	6
2.4 代码实现（部分代码）	13
2.5 运行结果	19
第 3 章 总结	22
参考文献	23

第 1 章 课程目的与要求

1.1 课程目的

JAVA 框架技术将该门课知识贯穿起来，巩固所学知识，进行小型的综合应用，对各知识点在整个应用的更加清晰，对所学知识达到融会贯通。学生在下述各方面的能力应该得到锻炼：

- (1) 巩固所学 java 基础知识、JSP、JavaBean, Servlet、Spring、SpringMVC、MyBatis 以及 Java 相关技术并了解软件开发的流程，形成系统的设计思想；
- (2) 通过学生查阅和搜集资料锻炼学生的自学能力，使学生能够理论联系实际；
- (3) 培养学生的沟通能力和团队合作精神；
- (4) 培养学生的工程化思想；
- (5) 培养学生总结归纳问题和表述问题的能力；
- (6) 提高学生对 Web 更深的理解。

1.2 实验环境

硬件要求能运行 Windows 操作系统的微机系统。JAVA 程序开发工具包 JDK1.8，WEB 服务器 Tomcat8.0，eclipse2019，MySQL5.0。

1.3 预备知识

熟悉 JAVA 语言、HTML、JSP 的基本知识及 Servlet、Spring、SpringMVC、MyBatis。

1.4 课程要求

- 1. 仔细分析设计题目，进行总体、详细设计，创建数据库，编写程序源代码。
- 2. 积极上机调试源程序，增强编程技巧与调试能力。
- 3. 认真书写实训预习报告, 实训说明书。
- 4. 遵守实训要求和机房管理制度，服从指导教师的安排，确保实训的顺利完成实训内容。

第 2 章 课程设计内容

2.1 需求分析

2.1.1 用户需求分析

随着科技的发展和生活水平的提高，人们对自身的健康状况也越来越重视。传统的手记个人的健康记录既麻烦又容易记错，特设计健康信息管理系统。健康管理平台的总目标是：在计算机网络，数据库和先进的开发平台上，利用现有的软件，配置一定的硬件，开发一个具有开放体系结构的、易扩充的、易维护的、具有良好人机交互界面的教师健康管理平台。

该系统主要针对的用户是社会所有人员，用户登陆后可以查看自己的体检记录和健康档案，还能记录各项健康指标的升降，进行留言以及修改个人信息。

2.1.2 功能需求分析

本系统的目的是实现个人健康信息管理系统的基本功能。健康信息系统为用户提供健康状况分析，健康档案管理，健康新闻、留言。满足不同用户需求；系统将用户分为：普通用户和系统管理员。

1. 普通用户能在本系统中进行注册登录后，查看健康趋势分析及预警，查看公告，个人的健康档案管理，体检记录管理，管理留言，修改个人资料。用户点击健康趋势分析，能看见自己的健康记录，包括名称、日期、血糖、总胆固醇、甘油三酯、低压、高压、心率、体温、呼吸次数、食欲、睡眠、健康状况、健康分析。点开公告查看，可查看公告列表。在健康档案管理可对档案记录进行添加。在体检记录管理可对体检记录进行添加。

2. 系统管理员可以登录该系对用户的档案，体检记录、留言、公告、管理员、用户进行增删改查

2.2 总体设计

根据健康信息管理系统的需求分析，确定此系统分为用户界面和管理员界面。

用户界面分为健康预警、健康趋势分析查看、公告查看、健康档案管理、体检记录管理、我的留言、修改个人资料等几个模块。

管理员通过输入的用户名和密码登陆后台管理区，可以对后台进行健康档案管理、体检记录管理、留言管理、公告管理和用户管理等几个模块的操作。在本系统中，总体设计包括：

- (1) 系统应包括两大部分：用户操作和管理员操作；
- (2) 一般用户应该可以在注册登录后，自由浏览界面，对自己的健康信息，体检记录进行查看和分析、进行留言，以及对自己的信息进行管理。
- (3) 系统管理员应该可以对档案和体检记录进行查看、修改、删除和添加操作，也可以对注册用户管理，同时还可以对相关留言评论进行管理。

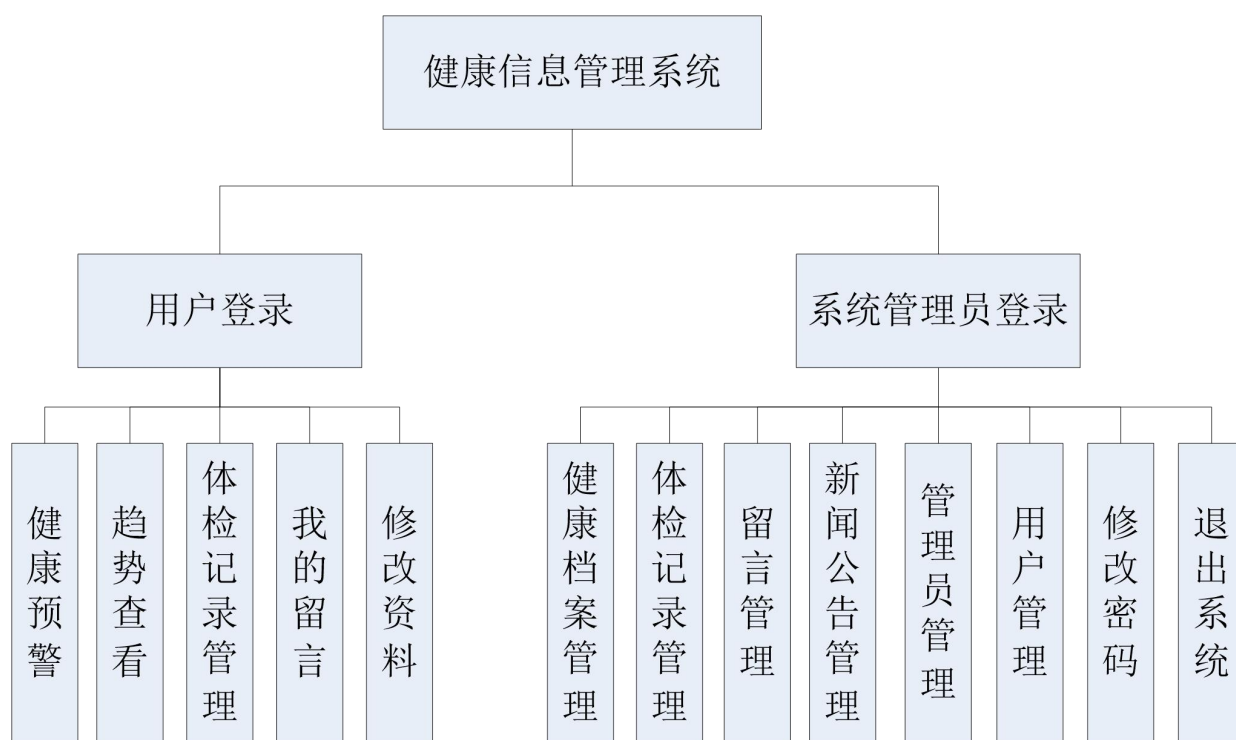


图 2-1 健康信息系统功能模块图

2.3 详细设计

2.3.1 数据库设计

该健康信息管理系统的数据库包含 6 个表：系统管理员密码表：admin、档案表 dangan、健康信息表 health、留言表 liuyan、新闻公告表 notice、用户表 user。

1. 系统管理员密码表：admin:用来保存管理员的信息。

表 2.1 admin

编号	字段名称	数据结构	说明
1	id	int	编号(主键)
2	adminname	varchar(50)	管理员姓名
3	pwd	varcahr(50)	密码

2. 档案表 dangan: 用来保存用户的健康信息档案。

表 2.2 dangan

编号	字段名称	数据结构	说明
1	id	int	编号(主键)
2	name	varchar(50)	名称
3	sdate	varcahr(50)	日期
4	xuetang	varcahr(50)	血糖
5	xuezhi	varchar(100)	血脂
6	ganyou	varcahr(50)	甘油
7	xueyal	varcahr(50)	低压
8	xueya	varchar(100)	高压
9	xinlv	varchar(100)	心率
10	tiwen	varchar(100)	体温
11	huxi	varchar(100)	呼吸次数
12	shiyu	varchar(100)	食欲
13	shui	varchar(100)	睡眠
14	uname	varchar(50)	用户
15	zt	varchar(50)	健康状态
16	yijian	varchar(200)	健康分析

3. 健康信息表 health: 显示用户的健康状态。

表 2.3 health

编号	字段名称	数据结构	说明
1	id	int	编号(主键)
2	sdate	varchar(50)	检查日期
3	name	varcahr(50)	名称
4	danwei	varchar(50)	检查医院
5	xiangmu	varchar(50)	检查项目
6	jieguo	varchar(50)	检查结果
7	cuoshi	varchar(50)	采取措施
8	beizhu	varchar(50)	备注
9	uname	varchar(50)	账号
10	zt	varchar(50)	预警状态
11	yijian	varchar(50)	健康状况

4. 留言表 liuyan, 用来保存用户在该系统发布的留言。

表 2.4 liuyan

编号	字段名称	数据结构	说明
1	id	int	编号(主键)
2	title	varchar(50)	留言标题
3	neirong	varchar(3000)	内容
4	huifu	varchar(250)	回复
5	uname	varchar(50)	用户
6	addtime	varchar(50)	时间

5. 新闻公告表 notice: 显示推荐的新闻。

表 2.5 notice

编号	字段名称	数据结构	说明
1	id	int	编号(主键)
2	title	varchar(150)	标题
3	pic	varchar(100)	图片

4	neirong	varchar(3000)	内容
5	addtime	varchar(50)	时间

6. 用户表 user: 显示有哪些人访问。

表 2.6 user

编号	字段名称	数据结构	说明
1	id	int	编号(主键)
2	name	varchar(50)	姓名
3	sex	varchar(20)	性别
4	sdate	varchar(50)	出生日期
5	tel	varchar(20)	联系电话
6	xueli	varchar(50)	学历
7	addr	varchar(50)	地址
8	username	varchar(50)	登录账号
9	pwd	varchar(50)	登录密码

2.3.2 程序结构设计

1. 持久对象层（也称持久层或持久化层）：该层由若干持久化类（实体类）组成。比如 Admin.java。

```
package com.entity;
public class Admin {
private Integer id;
private String adminname;
private String pwd;
public Integer getId()
{return id;}
public void setId(Integer id)
{this.id = id;}
public String getAdminname()
{return adminname;}
public void setAdminname(String adminname)
```

```
{this.adminname = adminname;}
public String getPwd()
{return pwd;}
public void setPwd(String pwd)
{this.pwd = pwd;}
@Override
public String toString() {
return "Admin [id=" + id + ", adminname=" + adminname + ", pwd=" + pwd + "];"
}
}
```

2. 数据访问层（DAO 层）：该层由若干 DAO 接口和 MyBatis 映射文件组成。比如 AdminMapper.java。

```
package com.mapper;
import java.util.List;
import org.apache.ibatis.annotations.Param;
import com.entity.Admin;
public interface AdminMapper {
//添加
public int insert(Admin admin);
//删除
public void delete(int id);
//根据id获取数据
public Admin get(int id);
//修改
public int update(Admin admin);
//列表
public List<Admin> list(@Param("where") String where);
//分页
public List<Admin> sealist(@Param("where") String where);
//统计
public int total();
}
```

3. 业务逻辑层（Service 层）：该层由若干 Service 接口和实现类组成。比如

```
AdminService.java。
package com.service;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.entity.Admin;
import com.mapper.AdminMapper;
@Service
public class AdminService{
    @Autowired
    AdminMapper adminMapper;
    public List<Admin> list(String where) {
        return adminMapper.list(where);
    }
    public List<Admin> sealist(String where) {
        // TODO Auto-generated method stub
        return adminMapper.list(where);
    }
    public int total() {
        return adminMapper.total();
    }
    public void insert(Admin c) {
        adminMapper.insert(c);
    }
    public void update(Admin c) {
        adminMapper.update(c);
    }
    public void delete(Admin c) {
        adminMapper.delete(c.getId());
    }
    public Admin get(int id) {
        // TODO Auto-generated method stub
        return adminMapper.get(id);
    }
}
```

4. Web表现层：该层主要包括Spring MVC中的Controller类和JSP (html) 页面。Controller类主要负责拦截用户请求，并调用业务逻辑层中相应组件的业务逻辑方法来处理用户请求，然后将相应的结果返回给JSP页面。比如AdminController.java。

```
package com.controller;

import java.util.List;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.ui.Model;
import com.entity.Admin;
import com.service.AdminService;
import com.util.Page;
import com.util.Pagesize;
import com.util.Pagination;
//告诉spring mvc这是一个控制器类
@Controller
//RequestMapping注解会将 HTTP 请求映射到 MVC 和 REST 控制器的处理方法上。
@RequestMapping("")
public class AdminController {
//它可以对类成员变量，方法及构造函数进行标注，完成自动装配的工作，通过
//@Autowired的使用来消除 set, get方法
@Autowired
AdminService adminService;
@Autowired
HttpServletRequest request;
//列表管理页面
@RequestMapping("adminlist")
public String listAdmin(Model model) {
String w="";
if (!"".equals(request.getParameter("adminname"))) &&
request.getParameter("adminname") !=null)
w=w+ " and adminname like '%" +request.getParameter("adminname")+"%' ";
List<Admin> cs= adminService.list(w);
```

```
for(int i=0;i<cs.size();i++)
{
Admin admin=(Admin)cs.get(i);
}
int index=0;
if(request.getParameter("index")==null)
{index=1;}
else
{index=Integer.parseInt(request.getParameter("index"));}
int fromIndex = (index - 1) * Pagesize.size;
int toIndex = Math.min(fromIndex + Pagesize.size, cs.size());
List<Admin> cs1 = cs.subList(fromIndex, toIndex);
Pagination p = new Pagination();//创建 分页对象
p.setIndex(index);//设置页数
p.setPageSize(Pagesize.size);
p.setTotle(cs.size());//设置总共的条数
p.setData(cs1);//设置数据
// 放入转发参数
model.addAttribute("cs", p);
model.addAttribute("page", p);
model.addAttribute("dlist", cs);
// 放入转发参数
return "pages/admin/adminlist";
}

@RequestMapping("adminadd")
public String addadmin(Model model) {
return "pages/admin/adminadd";
}

//添加数据
@RequestMapping("admininsert")
public String insertAdmin(Admin admin,Model model) {
String forword=request.getParameter("forword");
adminService.insert(admin);
model.addAttribute("msg","成功");
model.addAttribute("path",forword);
```

```

return "success";
}
//删除数据
@RequestMapping("admindel")
public String deleteAdmin(Admin admin, Model model) {
String forword=request.getParameter("forword");
adminService.delete(admin);
model.addAttribute("msg", "删除成功");
model.addAttribute("path", forword);
return "success";
} //提取要修改的数据
@RequestMapping("adminedit")
public String editAdmin(Model model, Admin admin) {
Admin c= adminService.get(admin.getId());
model.addAttribute("c", c);
return "pages/admin/adminedit";
}
//修改的数据
@RequestMapping("adminupdate")
public String updateAdmin(Admin admin, Model model, String
forword=request.getParameter("forword");
adminService.update(admin);
model.addAttribute("msg", "修改成功");
model.addAttribute("path", forword);
return "success";
}
//所有列表页
@RequestMapping("adminAll")
public String listAdminAll(Model model) {
String w="";
if ("".equals(request.getParameter("adminname"))) &&
request.getParameter("adminname") !=null)
w=w+ " and adminname like '%" +request.getParameter("adminname")+"%' ";
t<Admin> cs= adminService.list(w);
(int i=0;i<cs.size();i++)

```

```

{Admin admin=(Admin)cs.get(i);}
int index=0;
if(request.getParameter("index")==null)
{index=1;}
else
{index=Integer.parseInt(request.getParameter("index"));}
int fromIndex = (index - 1) * Pagesize.size;
int toIndex = Math.min(fromIndex + Pagesize.size, cs.size());
List<Admin> cs1 = cs.subList(fromIndex, toIndex);
Pagination p = new Pagination();//创建 分页对象
p.setIndex(index);//设置页数
p.setPageSize(Pagesize.size);
p.setTotle(cs.size());//设置总共的条数
p.setData(cs1);//设置数据
// 放入转发参数
model.addAttribute("cs", p);
model.addAttribute("page", p);
model.addAttribute("dlist", cs);
// 放入转发参数
return "pages/admin/adminAll";
}
//注册
@RequestMapping("adminreg")
public String insertreg(Admin admin,Model model){
String forword=request.getParameter("forword");
String w=" and username='"+request.getParameter("username")+"'";
List<Admin> cs= adminService.list(w);
if (cs.size()>0){
model.addAttribute("msg","用户名重复!请重试");
model.addAttribute("path", forword);
}
else{
adminService.insert(admin);
model.addAttribute("msg","成功");
model.addAttribute("path", forword);}
}

```

```

return "success";}
//修改个人资料
@RequestMapping("admininfo")
public String editAdmininfo(Model model,Admin admin){
Admin c= adminService.get(admin.getId());
model.addAttribute("c", c);
return "pages/admin/admininfo";
}
//保存修改个人资料数据
@RequestMapping("admininfosave")
public String Admininfosave(Admin admin,Model model){
String forword=request.getParameter("forword");
adminService.update(admin);
model.addAttribute("msg","修改成功");
    model.addAttribute("path",forword);
return "success";
}
}

```

5. 网站页面导航:



图 2.1 页面导航图

2.4 代码实现（部分代码）

1. UserMapper.java

```
package com.mapper;
```

```
import java.util.List;
import org.apache.ibatis.annotations.Param;
import com.entity.User;
public interface UserMapper {
//添加
public int insert(User user);
//删除
public void delete(int id);
//根据 id 获取数据
public User get(int id);
//修改
public int update(User user);
//列表
public List<User> list(@Param("where") String where);
//分页
public List<User> sealist(@Param("where") String where);
//统计
public int total();
public void updatesql(@Param("value")String value);
public void deletesql(@Param("value")String value);
}
```

2. Uer. java

```
package com.entity;
public class User {
private Integer id;
private String name;
private String sex;
private String sdate;
private String tel;
private String xueli;
private String addr;
private String username;
private String pwd;
public Integer getId()
{return id;}
```

```
public void setId(Integer id)
{this.id = id;}
public String getName()
{return name;}
public void setName(String name)
{this.name = name;}
public String getSex()
{return sex;}
public void setSex(String sex)
{this.sex = sex;}
public String getSdate()
{return sdate;}
public void setSdate(String sdate)
{this.sdate = sdate;}
public String getTel()
{return tel;}
public void setTel(String tel)
{this.tel = tel;}
public String getXueli()
{return xueli;}
public void setXueli(String xueli)
{this.xueli = xueli;}
public String getAddr()
{return addr;}
public void setAddr(String addr)
{this.addr = addr; }
public String getUsername()
{return username;}
public void setUsername(String username)
{this.username = username;}
public String getPwd()
{return pwd;}
public void setPwd(String pwd)
{this.pwd = pwd;}
@Override
```

```
public String toString() {
    return "User [id=" + id + ", name=" + name + ", sex=" + sex + ", sdate=" + sdate
    + ", tel=" + tel + ", xueli=" + xueli + ", addr=" + addr + ", username=" + username
    + ", pwd=" + pwd + "]";
}
}
```

3. UserService.java

```
package com.service;

import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.entity.User;
import com.mapper.UserMapper;

@Service
public class UserService{
    @Autowired
    UserMapper userMapper;

    public List<User> list(String where) {
        return userMapper.list(where);}

    public List<User> sealist(String where) {
        // TODO Auto-generated method stub
        return userMapper.sealist(where);}

    public int total() {
        return userMapper.total();}

    public void insert(User c) {
        userMapper.insert(c);}

    public void update(User c) {
        userMapper.update(c);}

    public void delete(User c) {
        userMapper.delete(c.getId());
    }

    public User get(int id) {
        // TODO Auto-generated method stub
        return userMapper.get(id);}

    public void updatesql(String value) {
```

```

userMapper.updateSql(value);}
public void deleteSql(String value) {
userMapper.deleteSql(value);
}
}

4. index.jsp
<%@ page language="java" import="java.util.*" pageEncoding="ISO-8859-1"%>
<%
String path = request.getContextPath();
String                               basePath                               =
request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()
+path+"/";
%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<base href="<%=basePath%>">
<title>My JSP 'index.jsp' starting page</title>
<meta http-equiv="pragma" content="no-cache">
<meta http-equiv="cache-control" content="no-cache">
<meta http-equiv="expires" content="0">
<meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
<meta http-equiv="description" content="This is my page">
<!--
<link rel="stylesheet" type="text/css" href="styles.css">
-->
</head>
<body>
<script>location.href="<%=path%>/pages/login.jsp"</script>
</body>
</html>

5. 配置文件 applicationContext.xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

```

```
xmlns:aop="http://www.springframework.org/schema/aop"
xmlns:tx="http://www.springframework.org/schema/tx"
xmlns:jdbc="http://www.springframework.org/schema/jdbc"
xmlns:context="http://www.springframework.org/schema/context"
xmlns:mvc="http://www.springframework.org/schema/mvc"
xsi:schemaLocation="
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.0.xsd
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
http://www.springframework.org/schema/jdbc
http://www.springframework.org/schema/jdbc/spring-jdbc-3.0.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-3.0.xsd
http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-3.0.xsd
http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc.xsd">
<!-- 通过注解，将 Service 的生命周期纳入 Spring 的管理 -->
<context:annotation-config />
<context:component-scan base-package="com.service" />
<!-- 配置数据源 -->
<bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource"
init-method="init" destroy-method="close">
<!-- 基本属性 url、user、password -->
<property name="url" value="jdbc:mysql://localhost:3306/jiankang?characterEncoding=UTF-8" />
<property name="username" value="root" />
<property name="password" value="123" />
<property name="driverClassName" value="com.mysql.jdbc.Driver" />
<!-- 配置初始化大小、最小、最大 -->
<property name="initialSize" value="3" />
<property name="minIdle" value="3" />
<property name="maxActive" value="20" />
<!-- 配置获取连接等待超时的时间 -->
```

```
<property name="maxWait" value="60000" />
<!-- 配置间隔多久才进行一次检测，检测需要关闭的空闲连接，单位是毫秒 -->
<property name="timeBetweenEvictionRunsMillis" value="60000" />
<!-- 配置一个连接在池中最小生存的时间，单位是毫秒 -->
<property name="minEvictableIdleTimeMillis" value="300000" />
<property name="validationQuery" value="SELECT 1" />
<property name="testWhileIdle" value="true" />
<property name="testOnBorrow" value="false" />
<property name="testOnReturn" value="false" />
<!-- 打开 PSCache，并且指定每个连接上 PSCache 的大小 -->
<property name="poolPreparedStatements" value="true" />
<property name="maxPoolPreparedStatementPerConnectionSize" value="20" />
</bean>
<!-- 扫描存放 SQL 语句的.xml-->
<bean id="sqlSession" class="org.mybatis.spring.SqlSessionFactoryBean">
<property name="typeAliasesPackage" value="com.entity" />
<property name="dataSource" ref="dataSource"/>
<property name="mapperLocations" value="classpath:com/mapper/*.xml"/>
</bean>
<!-- 扫描 Mapper，并将其生命周期纳入 Spring 的管理 -->
<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
<property name="basePackage" value="com.mapper"/>
</bean>
</beans>
```

2.5 运行结果

1. 管理员登录界面：管理员在该页面输入用户名和密码，选择管理员选项即可登录。



图 2.2 管理员登录界面

4. 登录成功界面：该页面是输入用户名和密码后会弹出来的窗口，如果用户名和密码正确则显示登录成功，否则显示登录失败。



图 2.5 登录成功界面

5. 健康档案界面：在该页面可查看用户在该系统内存储的健康档案信息。



图 2.6 健康档案界面

6. 体检记录界面：可在该页面查看用户的体检记录。



图 2.7 体检记录界面

第3章 总结

该系统是健康信息管理系统，主要为用户提供平台，对自己的健康有个全面的认知。

该系统的优点在于将系统分为管理员登录和用户登录两大模块，分工明确，用户登录模块是给用户提供更方便的服务，记录一些个人的健康档案、体检记录以及对自己的情况进行简单地分析，还可以浏览一些健康、养生的新闻，并在其下方留言。而管理员界面是管理员管理，更新信息，管理留言和用户信息等。

该系统页面简洁大方，操作起来也十分上手，对于不会操作的老年人也很友好。该系统使用的是 MVC 架构模式，能对系统的功能进行分离，对系统分成五个层次。其中持久对象层是由若干持久化类（实体类）组成。数据访问层由若干 DAO 接口和 MyBatis 映射文件组成。业务逻辑层由若干 Service 接口和实现类组成。Web 表现层主要包括 Spring MVC 中的 Controller 类和 JSP (html) 页面。其中 Controller 是应用程序中处理用户交互的部分是系统的控制层，用户通过系统对数据的访问通过控制层对数据的控制之后通过模型处理再返回数据给控制层之后反馈给用户就能获取信息。

通过编写健康信息管理系统我能更好的了解编程语言的强大，通过一些设计模式能使我们编写的系统更能更加强大和兼容性更加完美，可拓展性更强。一款强大的系统应该具有拓展性好，兼容性强等功能。我们进行系统的设计时候多使用一些现在流行的设计模式可以使系统更加完善。

在今后的学习生活中，我会更加完善自己，巩固知识，学好 javaEE 技术，为将来的工作打好基础。

参考文献

1. 郑阿奇. java EE 实用教程（第 2 版），电子工业出版社，2015
2. 古乐声等. JavaWeb 程序设计与项目实践[M]. 北京:电子工业出版社. 2011.
3. 李刚. 轻量级 Java EE 企业应用实战[M]. 北京:电子工业出版社. 2012.
4. 乔岚. 基于 MyBatis 和 Spring 的 JavaEE 数据持久层的研究与应用[J]. 宁夏财经职业技术学院. 2017
5. 陈永政. JavaEE 框架技术: SpringMVC+Spring+MyBatis[M]. 西安: 西安电子科技大学出版社. 2017
6. 王爱华. HTML+CSS+JavaScript 网页制作简明教程[M]. 北京: 清华大学出版社. 2014.
7. 赵锋. 网页设计与制作. 北京: 清华大学出版社. 2013
8. 黄缙华. MySQL 入门很简单. 北京: 清华大学. 2011