**《Web编程技术》报告**

**基于servlet的电脑故障与解决方法信息共享系统的设计与实现**

**学 号：**

**专业班级：**

**姓 名：**

**计算机科学与工程学院**

**目录**

[一、设计目的 5](#_Toc19423)

[二、需求分析 6](#_Toc2164)

[1、技术需求 6](#_Toc21716)

[1) Maven 6](#_Toc1543)

[2) MySQL数据库与JDBC 6](#_Toc6505)

[2、基本要素需求 7](#_Toc27779)

[1）数据要素 7](#_Toc1253)

[2）故障记录系统要素 7](#_Toc6748)

[3）用户登录系统要素 7](#_Toc17168)

[三、 系统结构与关联 8](#_Toc16085)

[1、 前后端结构 8](#_Toc5718)

[2、 数据库结构 9](#_Toc28667)

[四、 界面展示 11](#_Toc27403)

[五、 部分特色的核心代码 16](#_Toc25364)

[六、 设计总结 24](#_Toc20734)

# 一、设计目的

在学校的志愿活动中，我发现有许多同学的电脑经常出现故障，但是苦于没有网络上相关资料过于繁多而且不知道如何筛选对自己设备上出现的故障有效的解决方式。由此产生了制作这个电脑故障与解决方案的分享交流网站。

该系统的基本作用是系统的给用户提供详细的故障产生环境（比如说系统环境、设备信号、日期等），并提供电脑故障的解决方案，类似与医院中使用的病历一样，需要记录故障产生的各种相关因素。相较于在搜索引擎上直接搜索，该系统可以更加直观的提供故障发生的相关条件，帮助用户更加高效的筛选对自己有用的信息。

对于已经完成了故障处理的用户，也可以通过该系统记录故障产生的具体表现，将自己的解决方法记录下来于其他人共同交流。这样无论是自己还是其他人再次出现类似的故障时，可以参考记录下来的故障与解决方法，这样就更加高效的帮助所有人高效并且有效的解决自己遇到的电脑故障。

选择这种“取之于人，用之于人”的用户自行提交所需数据的社区式信息共享也是因为遇到的故障多种多样，不可能在一开始就设置好所有可能的故障情况。并且由于更加系统的整理了故障出现的环境，可以更有方向性的去寻求帮助。举个例子，当某个型号的机器在都出现过相似的问题，可以去早厂商寻求帮助；或者大量设备频繁卡顿是在推出了某个系统更新的这段时间之后，就可以向系统反馈该类问题，用户也可以知道是因为系统导致的共性问题而选择回滚系统之类有方向的修复举措。

# 二、需求分析

## 1、技术需求

1. Maven

Maven是一个主要用于Java项目的自动化构建工具。Maven与Apache Ant工具具有相似的目的，但是他们是基于不同的理念并且以不同的方式来工作。Ant还可以用来构建和管理C#，Ruby，Scala以及其他的语言开发的项目，但是Maven原生并不支持这些。Maven项目由Apache Software Fundation托管，之前是Jakarta Project项目的一个部分。

Maven使用XML文件来描述需要构建的软件项目、项目的依赖模块和组件、构建顺序、目录结构以及需要的插件。他具有预定义的目标来实现确切的任务，比如编译源码并打包。

Maven动态地从一个或多个库中下载Java库和Maven插件，比如Maven 2 Central Repository，并且将下载的内容保存到本地的缓存中。这个本地的下载缓存可以被本地项目更新，公共的库也可以被更新。

Maven使用基于插件的体系来构建，他允许通过标准的输入控制任何的应用。理论上说，这将允许任何人为任何语言来在这个平台上编写插件。实际上，支持和使用除Java外其它语言的插件数量已经微乎其微。目前，只有支持.NET框架以及一个C/C++的原生插件还在被维护。

项目对象模型（POM）：一个项目对象模型提供一个项目的所有配置属性。一般配置包括项目名称，owner和它的依赖。也可以通过插件，配置构建过程的各个阶段。比如说，用户可以配置编译插件，让它使用Java 1.5来编译，或者指定在一些单元测试失败以后，依旧打包工程。大的项目应该分成几个模块或者子项目，每个模块拥有自己的POM。用户可以编写一个根POM，通过这个根POM来使用一条命令来编译所有的模块。POM也可以从其它POM文件继承配置信息。所有的POM文件默认继承自Super POM。Super POM提供默认的配置，比如默认源代码目录、默认插件，等等。

1. MySQL数据库与JDBC

由于该系统中会产生大量的数据整合与交互，同时动态的加载到jsp构建的网页上，所以需要一个高效的数据库系统。选用了基于MySQL数据库建立病历管理系统，设计并建立了所需的数据库用于用户的交互。

## 2、基本要素需求

1）数据要素

初始时需要一定量的已有故障与解决方式来进行网站的基础服务，在之后通过用户的提交来逐渐丰富故障记录的数据库。

为实现用户需求，至少需要记录故障的具体形式与解决方法，同时也是最为关键的信息。除此之外每一个故障需要记录多种故障产生的环境条件，在这里选取最常见的故障产生时间（可能涉及某个更新的推出）、设备型号（可能涉及某些型号的设计缺陷）与系统（可能涉及不同系统间的差异）进行记录。

1. 故障记录系统要素

要符合实际要求，因为该动态网站同时提供查询记录和提交记录两个功能，需要分别对应两个大的模块，本别实现根据用户输入的故障症状对现有数据库的模糊搜索的功能，以及用户提交故障记录的功能。

1. 用户登录系统要素

该网站为了考虑安全性，设置了用户登录系统，在进入主页之前进行登录操作。未登录用户可以查询故障记录但是不能进行提交，尽量避免虚假或错误的信息被录入。登录用户则在查询的功能之外，可以进行提交的功能，帮助已有记录的丰富。

# 系统结构与关联

项目遵循了软甲开发的三层架构的设计，表现层（jsp与css等前端内容）用于展示数据，业务层（servlet）用于处理业务需求，持久层（MySQL的交换类型database）用于完成与数据库的交互。整个系统可分为前后端结构与数据库结构。具体阐述如下：

## 前后端结构

表现层包含每个界面的jsp文件。总共6个页面，分别为首页index.jsp，查询界面search.jsp，上报界面report.jsp，登录注册界面login.jsp，查询结果界面result.jsp，单个记录详细结果界面detail.jsp。jsp文件都存放在./web路径中，对应的css文件存放在./web/css路径中。

业务层包含实现所有功能的servlet文件。总共5个servlet，分别为根据关键词对所有记录进行模糊搜索功能的SearchServlet，查看单个记录全部信息功能的GetProblemServlet，提交用户填写的记录到数据库功能的ReportServlet，验证用户登录信息功能的SignInServlet，将新用户的用户信息提交到数据库的SignUpServlet。servlet对应的java文件都在./src/servlet 路径中。

持久层包含与数据库交互的内容。总共3个，本别为对记录进行数据库操作的ProblemDB.java和对用户进行数据库操作的UserDB.java以及一个完成了对链接数据库、关闭数据库进行封装的基类Database.java。ProblemDB和UserDB都继承Database。对应的java文件都在./src/database 路径中

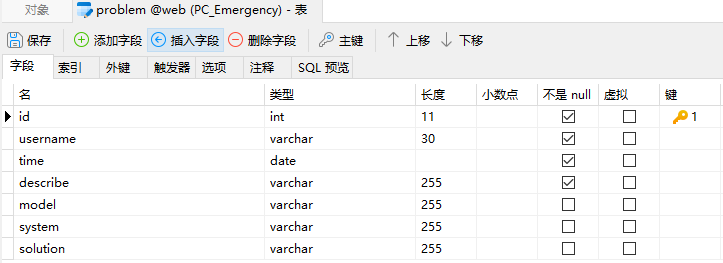
具体的数据流向和跳转控制如下图所示：

## 未命名文件

## 数据库结构

Web数据库中存在两个表，分别为储存电脑故障记录的problem表和存放用户数据的user表。

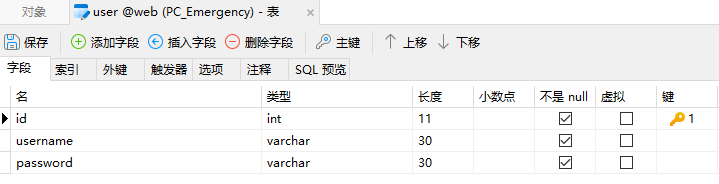
problem表的字段如下：



每个字段的释义如下表:

|  |  |
| --- | --- |
| id | 故障记录的id，由MySQL自动递增生成 |
| username | 提供该条记录的用户名称 |
| time | 提供该条记录时的日期 |
| describe | 该条记录对故障的详细描述 |
| model | 发生该故障的机器型号 |
| system | 发生该故障的电脑的系统类型 |
| solution | 该故障的解决方式 |

user表的字段如下：



每个字段的释义如下表:

|  |  |
| --- | --- |
| id | 该用户的id，由MySQL自动递增生成 |
| username | 该用户的用户名 |
| password | 该用户的密码 |

# 界面展示

由于文档格式限制无法演示动画效果，动画效果通过视频录制展示。报告中展示内容为静态效果。

首页展示







搜索界面



无搜索结果时的搜索结果界面：



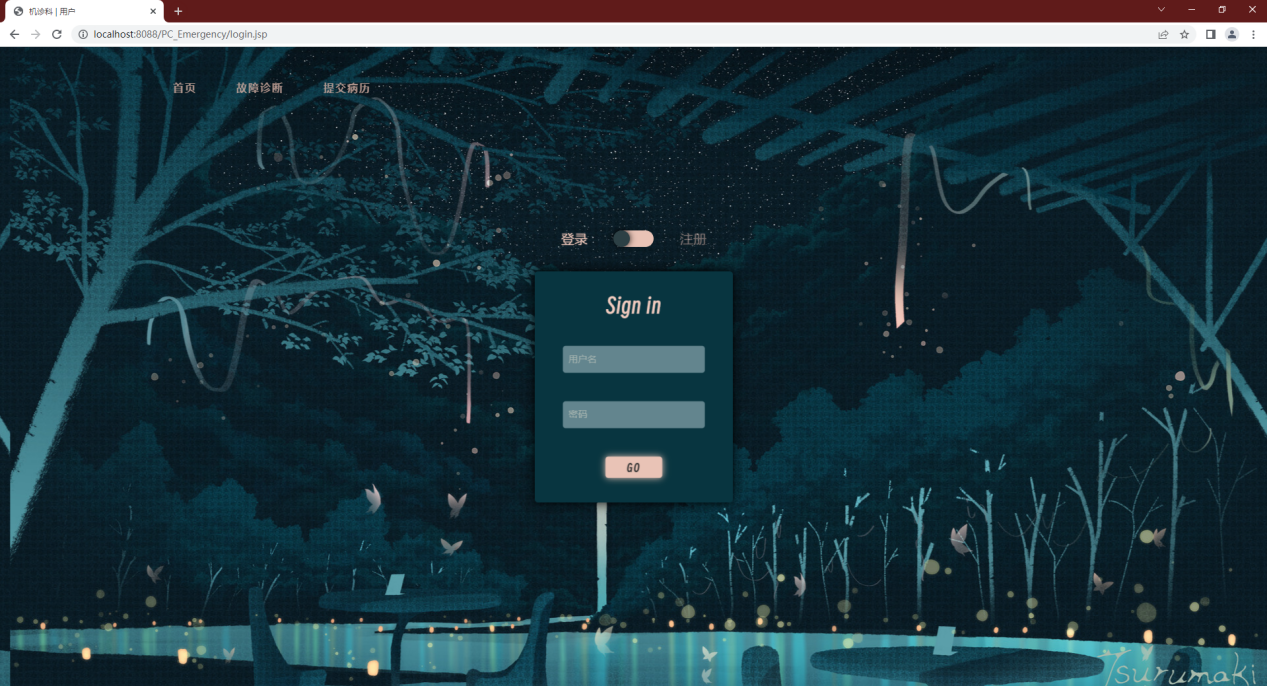
有搜索结果的搜索结果界面：



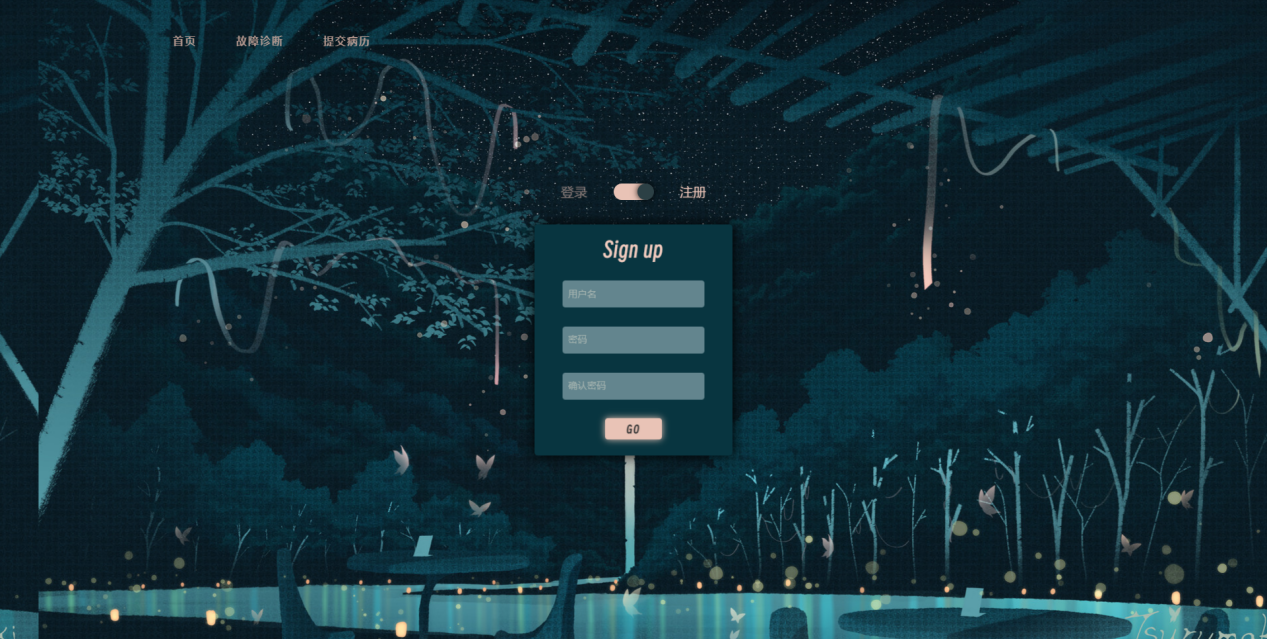
单击某一个结果后进入对应的详情界面展示：



登录界面展示：



注册界面展示：



登录后导航栏中的登录部分变为根据当前时间进行问好：

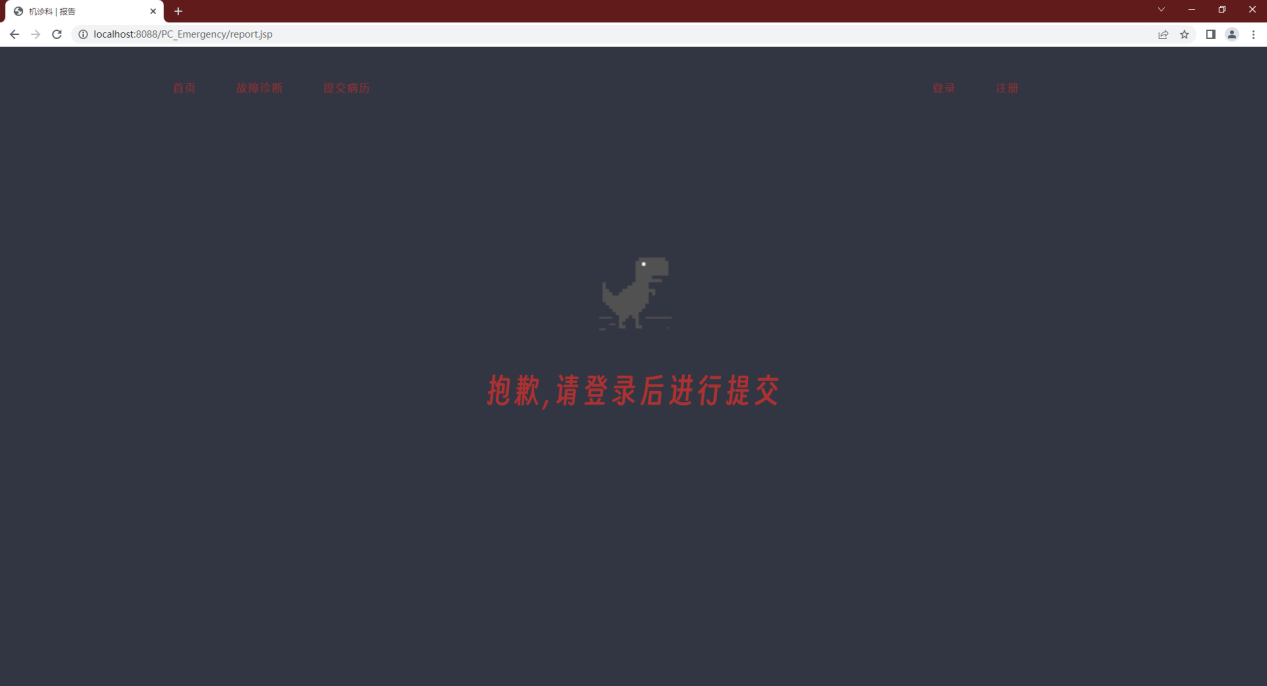
登陆前



登录后



提交故障记录（病历）的界面，未登录时无法进行病历的提交



提交故障记录（病历）登录后的界面



# 部分特色的核心代码

1. 首页index.jsp：
2. 标题的3D阴影效果的css代码：

/\*文字阴影立体效果\*/

.text-shadow-pop {

text-shadow: 0 0 rgb(38, 48, 69);

transition-duration: 0.5s;

}

.text-shadow-pop:hover {

text-shadow: 6px 6px rgb(38, 48, 69);

transition-duration: 1s;

}

/\*背景加强移动颜色\*/

.cross\_highlight {

position: relative;

z-index: 1;

}

.cross\_highlight::before {

content: '';

position: absolute;

z-index: -1;

top: 0;

bottom: 0;

left: -0.3em;

right: -0.3em;

background-color: rgb(189, 134, 105);

transform-origin: center right;

transform: scaleX(0) scaleY(0.6);

transition: transform 0.2s ease-in-out;

}

.cross\_highlight:hover::before {

transform: scaleX(1) scaleY(0.6);

transform-origin: center left;

}

1. 文字和背景差速滚动的部分html和css代码：（将文字利用translateZ(1px)与透视效果结合实现文字移动速度比背景快的效果）

html部分：

<div class="page">

......

<!-- 病例版块 -->

<div class="library\_background background"></div>

<!-- 病例子标题 -->

<div class="library\_title">

找找病历?

</div>

<!-- 查询部分 -->

<div class="library\_search">

<div class="library\_search\_box">

<div class="library\_search\_info">

请各位有

<span class="hover\_recolor\_light\_blue"> 蓝屏症状</span>、

<span class="hover\_recolor\_black"> 黑屏症状</span>、

<span class="hover\_flicker"> 闪烁症状</span>

等常见病的患机先进行自我诊断

</div>

<div class="library\_search\_input\_box">

<form method="post" action="./search">

<input type="text" name="keyword" required>

<span>症状是……</span>

</form>

</div>

</div>

</div>

......

CSS部分：

......

.page {  
 height: 100vh;  
 overflow-x: hidden;  
 perspective: 3px;  
 background-color: rgb(228, 223, 170);  
 }

......

.library\_background {

display: flex;

top: 10vh;

height: 110vh;

background-color: rgb(189, 134, 105);

background-image: url(../resourse/pictures/pixel\_TV.png);

}

.library\_title {

display: flex;

justify-content: left !important;

transform: translateY(-65vh) translateX(22%) translateZ(1px);

color: rgb(85, 39, 0);

font-family: "SmileySans", sans-serif;

font-size: 100px;

}

.library\_search {

display: flex;

justify-content: left !important;

transform: translateY(-62vh) translateX(22%) translateZ(1px);

height: 150px;

}

......

1. 导航栏滚动时根据不同的背景进行变色的js脚本。实际上是判断背景图片的元素限度浏览器左上角的距离来进行判断，并通过修改导航栏的class类型来实现的变色。

<!-- 导航栏变色效果 -->

<script type="text/javascript">

let page = document.querySelector(".page")

page.onscroll = function () {

let header = document.querySelector("header");

let main\_background = document.querySelector(".main\_background");

let library\_background = document.querySelector(".library\_background");

let appointment\_background = document.querySelector(".appointment\_background");

header.classList.toggle("default", main\_background.getBoundingClientRect().top > -50);

header.classList.toggle("main\_color", main\_background.getBoundingClientRect().top < -50 && main\_background.getBoundingClientRect().bottom > 20);

header.classList.toggle("library\_color", library\_background.getBoundingClientRect().top < 0 && library\_background.getBoundingClientRect().bottom > 20);

header.classList.toggle("appointment\_color", appointment\_background.getBoundingClientRect().top < 0 && appointment\_background.getBoundingClientRect().bottom > 10);

}

</script>

1. 根据时间对用户进行问好

<!-- 问候语根据时间修改 -->

<script>

// 1.获取元素

let greeting = document.querySelector('.greeting');

// 2. 得到当前的小时数

let hours = new Date().getHours();

if (hours <= 6) {

greeting.innerHTML = "晚上好，"

} else if (hours <= 11) {

greeting.innerHTML = "早上好，"

} else if (hours <= 14) {

greeting.innerHTML = "中午好，"

} else if (hours <= 16) {

greeting.innerHTML = "下午好，"

} else if (hours <= 18) {

greeting.innerHTML = "傍晚好，"

} else {

greeting.innerHTML = "晚上好，"

}

</script>

1. 关键字模糊查询
   1. 搜索页面 search.jsp中输入搜索关键词的jsp部分，使用form标签来给在/search的SearchServlet进行数据传递：

<form method="post" action="./search">

<input type="text" name="keyword" required>

<span>描述一下症状吧</span>

</form>

* 1. SearchServlet中处理关键字的部分，将调用持久层的problemDB中的searchProblem方法与数据库通信，将所有包含输入的关键字的结果储存在一个ArrayList中，并附加到Session中，跳转到结果显示界面。

@WebServlet(name = "SearchServlet", urlPatterns = "/search")

public class SearchServlet extends HttpServlet {

@Override

protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

resp.setHeader("content-type", "text/html;charset=UTF-8");

resp.setCharacterEncoding("utf-8");

req.setCharacterEncoding("utf-8");

String keyword = req.getParameter("keyword");

try {

ProblemDB problemDB = new ProblemDB();

ArrayList<ProblemData> problems = problemDB.searchProblem(keyword);

req.getSession().setAttribute("problems", problems);

req.getSession().setAttribute("keyword", keyword);

resp.sendRedirect("result.jsp");

problemDB.close();

} catch (ClassNotFoundException | SQLException e) {

throw new RuntimeException(e);

}

}

@Override

protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

doPost(req, resp);

}

}

其中ProblemData是自定义的用来储存故障记录的一个模型类，它的部分定义如下：

public class ProblemData {

private Integer id;

private String username;

private Date time;

private String describe;

private String model;

private String system;

private String solution;

......

}

* 1. SearchServlet调用的记录数据库交互类ProblemDB.java中的部分代码如下：

public class ProblemDB extends Database {

public ProblemDB() throws SQLException, ClassNotFoundException {

link();

}

/\*\*  
 \* 连接数据库  
 \*/  
protected void link() throws ClassNotFoundException, SQLException {  
 Class.forName("com.mysql.cj.jdbc.Driver");// 注册驱动  
 this.connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/web?serverTimezone=GMT%2B8", "root", "123456");// sql数据库链接  
}  
/\*\*  
 \* 关闭数据库链接  
 \*/  
public void close() throws SQLException {  
 connection.close();  
}

/\*\*

\* 根据关键字搜索匹配项

\*/

public ArrayList<ProblemData> searchProblem(String keyword) throws SQLException {

ArrayList<ProblemData> problems = new ArrayList<>();

keyword = "%" + keyword + "%";

PreparedStatement prep = connection.prepareStatement("select \* from web.problem where problem.describe like ?");

prep.setString(1, keyword);

prep.execute();

ResultSet resultSet = prep.executeQuery();

while (resultSet.next()) {

int id = resultSet.getInt("id");

String username = resultSet.getString("username");

Date time = resultSet.getDate("time");

String describe = resultSet.getString("describe");

String model = resultSet.getString("model");

String system = resultSet.getString("system");

String solution = resultSet.getString("solution");

problems.add(new ProblemData(id, username, time, describe, model, system, solution));

}

return problems;

}

......

}

* 1. 结果页面 result.jsp中展示搜索结果的jsp部分，使用c:forEach循环标签来对应输出每一个搜索结果：

<!-- 结果展示 -->  
<%  
 ArrayList<ProblemData> problems = (ArrayList<ProblemData>) request.getSession().getAttribute("problems");  
 request.setAttribute("problems",problems);  
 if (problems == null || problems.size() == 0) {  
%>  
<%--没有搜索结果--%>  
<div class="search\_result">  
 <div class="no\_result">  
 <div class="icon"></div>  
 <div class="text">抱歉,<br>暂时没有相关记录</div>  
 </div>  
</div>  
<%} else {%>  
<%--有搜索记录--%>  
<div class="search\_result">  
 <div class="search\_result\_box">  
 <div class="subtitle">  
 <ul>  
 <li class="problem\_describe">问题描述</li>  
 <li class="problem\_model">电脑型号</li>  
 <li class="problem\_system">系统版本</li>  
 </ul>  
 </div>  
 <c:forEach items="${requestScope.problems}" var="now">  
 <div class="signal">  
 <ul>  
 <li class="problem\_describe"><a  
 href="./getProblem?problem=${now.getId()}" target="\_blank">${now.getDescribe()}</a>  
 </li>  
 <li class="problem\_model"><a  
 href="./getProblem?problem=${now.getId()}" target="\_blank">${now.getModel()}</a>  
 </li>  
 <li class="problem\_system"><a  
 href="./getProblem?problem=${now.getId()}" target="\_blank">${now.getSystem()}</a>  
 </li>  
 </ul>  
 </div>  
 </c:forEach>  
 </div>  
</div>  
<% } %>

更加详细的功能演示在视频中体现。

# 设计总结

刚开始制作网页的时候，由于并不理解像是flex布局等等内容，所有功能都是想当然的，只是考虑了想要实现什么功能，但没有考虑具体改怎么实现，导致前期写好的网页有着奇奇怪怪的问题，并且完全没有考虑窗口大小导致的缩放问题。后来重新找资料重新学习了一下html和各类的布局标签，再同样的从头学习了一遍css后才做出来整体感觉差不多的效果。就是因为对于各种属性和标签不够了解，要不断地进行实践，总结经验。

在制作的期间也不断地去学习新的知识，也见识到了许多设计师设计的css动画与布局，让我了解到css动画竟然可以做出这么多效果。导致后来有很长一段时间都在设计动画和css效果，到最后整个系统虽然页面很好看，但是后端的交互内容就有些单薄。

开始编写后端之后，我发现后端的内容和前端完全不一样。刚开始，我并不熟悉后端相关的知识，所以在刚开始制作时屡屡碰壁，后来经过老师的讲解还有和同学的交流中逐渐理解了servlet的工作原理，由于学习过数据库的知识,查阅相关资料后也慢慢能熟练的通过使用JDBC将sql查询语句与java结合在一起。

本次课设中还存在一些不足和遗憾。比如没有尝试去使用框架来进行搭建。正如老师上课时提到的，现在公司中构建网页一般都会使用各式各样的框架来协助开发，而不是像我在本次课设中一样完全裸代码编辑。还有不足是没能完全挖掘用户系统，还可以添加管理员后台之类的功能来进一步切合实际的场景的需求。

总的来说，在完成这次课设时，我能感觉我从对web开发一无所知到掌握各模块的使用方法来完成目标的学习与成长。从最初的只是感兴趣的零基础到现在能够独立完成一个项目，让我对整个web编程的知识框架有了完整的认知。感谢老师的认真讲解，无私地为我们分享相关资料，用简单易懂的例子和生动的讲解带领我们走进Web的大门。