**实验3 简历查询与菜单权限设置**

## **实验目标**

本实验任务实现“Q-ITOffer”锐聘网站后台管理系统的简历分页查询、简历详情展示、依据用户角色的操作菜单展示功能：

* 【实验6-1】 简历列表分页查询。
* 【实验6-2】 简历详情展示。
* 【实验6-3】 依据用户角色的操作菜单展示。

## **【实验6-1】简历列表分页查询**

### 知识点回顾

* EL表达式中与范围有关的隐含对象
* EL表达式中pageContext隐含对象
* JSTL核心标签库

### 功能描述

实现“Q-ITOffer”锐聘网站后台管理系统的简历列表分页查询功能。

### 实验分析

1. 首先通过菜单目录发起一个分页查询简历列表数据的Servlet请求，通过此请求查询显示出第一页的简历列表数据。
2. 创建简历信息请求处理Servlet文件ResumeServlet.java，在此Servlet中调用简历数据库操作类ResumeDAO.java获得请求页分页信息，将请求页分页信息封装到一个通用的分页JavaBean类PageBean的对象中，然后将此分页JavaBean对象存入请求域属性响应回简历分页列表页面resumeList.jsp。
3. 创建简历信息数据库操作DAO类ResumeDAO.java，实现对简历总记录数的查询以及对请求页码的数据查询。
4. 创建简历基本信息JavaBean文件ResumeBasicinfo.java用来封装从数据库中查询到的简历基本信息数据。
5. 创建通用分页JavaBean文件PageBean.java，实现对每页显示记录数、当前页码、总页数、总记录数、请求页数据、是否有上一页、是否有下一页的封装。
6. 创建简历分页列表JSP页面resumeList.jsp，从请求域中获取请求页简历数据，进行遍历显示。resumeList.jsp页面运行效果如图6- 2所示。
7. 上述各资源组件间的关系如图6- 1所示。



图6- 1 企业列表功能组件关系图

### 实验步骤

1. 在Eclipse工具中创建Java Web项目“Q\_ITOffer\_Experiment07”，导入【实验五】中的所有资源，使每个后续实验都能够保证前期实验的所有功能的实现。
2. 更改后台主界面左侧目录文件left.html中建立查询的链接地址，设定请求地址为ResumeServlet，请求参数为type=list。

**【代码6- 1】left.html**

<a href="ResumeServlet?type=list" target="rightFrame">简历查询</a>

1. 在项目com.qst.itofferbacker.servlet包下创建ResumeServlet.java文件，对简历分页查询请求的处理代码如下所示。

**【代码6- 2】ResumeServlet.java**

package com.qst.itofferbacker.servlet;

import com.qst.itofferbacker.javabean.ResumeBasicinfo;

import com.qst.itofferbacker.dao.ResumeDAO;

import com.qst.itofferbacker.javabean.PageBean;

/\*\*

\* 简历信息管理Servlet

\* @公司 青软实训

\* @作者 fengjj

\*/

@WebServlet("/ResumeServlet")

public class ResumeServlet extends HttpServlet {

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

this.doPost(request, response);

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// 设置请求字符编码

request.setCharacterEncoding("UTF-8");

String type = request.getParameter("type");

if("list".equals(type)){

// 获取请求页码

int pageNo =

Integer.parseInt(request.getParameter("pageNo")==null?"1":

request.getParameter("pageNo"));

ResumeDAO dao = new ResumeDAO();

// 查询总记录数

int recordCount = dao.getRecordCount();

// 查询请求页码数据

List<ResumeBasicinfo> list = dao.getPageList(pageNo, 10);

// 将分页信息封装到PageBean对象中

PageBean<ResumeBasicinfo> pageBean =

new PageBean<ResumeBasicinfo>();

pageBean.setPageNo(pageNo);

pageBean.setRecordCount(recordCount);

pageBean.setPageData(list);

// 将分页数据对象PageBean对象存入请求域属性中

request.setAttribute("pageBean", pageBean);

// 将请求转发到简历列表页面

request.getRequestDispatcher("manage/resumeList.jsp")

.forward(request, response);

}

}

}

1. 在项目com.qst.itofferbacker.dao包下创建简历信息数据库操作DAO类ResumeDAO.java，增加查询简历总记录数的方法getRecordCount()、查询请求页简历信息的方法getPageList()。

**【代码6- 3】ResumeDAO.java**

package com.qst.itofferbacker.dao;

import com.qst.itofferbacker.javabean.ResumeBasicinfo;

import com.qst.itofferbacker.javabean.User;

import com.qst.itofferbacker.util.DBUtil;

/\*\*

\* 简历信息数据库操作类

\* @公司 青软实训

\* @作者 fengjj

\*/

public class ResumeDAO {

/\*\*

\* 查询所需分页的总记录数

\* @param pageSize

\* @return

\*/

public int getRecordCount() {

int recordCount = 0;

Connection conn = DBUtil.getConnection();

PreparedStatement pstmt = null;

ResultSet rs = null;

try {

String sql = "SELECT count(\*) FROM tb\_resume\_basicinfo";

pstmt = conn.prepareStatement(sql);

rs = pstmt.executeQuery();

if (rs.next())

recordCount = rs.getInt(1);

} catch (Exception e) {

e.printStackTrace();

} finally {

DBUtil.closeJDBC(rs, pstmt, conn);

}

return recordCount;

}

/\*\*

\* 查询请求页的简历数据

\* @return

\*/

public List<ResumeBasicinfo> getPageList(int pageNo, int pageSize) {

// 定义本页记录索引值

int firstIndex = pageSize \* (pageNo-1);

List<ResumeBasicinfo> list = new ArrayList<ResumeBasicinfo>();

Connection connection = DBUtil.getConnection();

if (connection == null)

return null;

PreparedStatement pstmt = null;

ResultSet rs = null;

try {

pstmt = connection

.prepareStatement("SELECT \* FROM ( SELECT a.\* , ROWNUM rn FROM ( "

+ "SELECT basicinfo\_id,realname,telephone,email,

job\_experience,job\_intension "

+ "FROM tb\_resume\_basicinfo ) a WHERE ROWNUM<=? ) WHERE rn>? ");

pstmt.setInt(1, firstIndex+pageSize);

pstmt.setInt(2, firstIndex);

rs = pstmt.executeQuery();

while (rs.next()) {

ResumeBasicinfo resume = new ResumeBasicinfo();

resume.setBasicinfoId(rs.getInt("basicinfo\_id"));

resume.setRealName(rs.getString("realname"));

resume.setTelephone(rs.getString("telephone"));

resume.setEmail(rs.getString("email"));

resume.setJobExperience(rs.getString("job\_experience"));

resume.setJobIntension(rs.getString("job\_intension"));

list.add(resume);

}

} catch (Exception e) {

e.printStackTrace();

} finally {

DBUtil.closeJDBC(rs, pstmt, connection);

}

return list;

}

}

1. 在项目com.qst.itofferbacker.javabean包下创建简历基本信息JavaBean文件ResumeBasicinfo.java用来封装从数据库中查询出的简历信息，JavaBean中属性与数据库中简历基本信息表TB\_RESUMEBASICINFO中的字段相对应，具体实现代码如下所示。

**【代码6- 4】ResumeBasicinfo.java**

package com.qst.itofferbacker.javabean;

import java.util.Date;

/\*\*

\* 简历信息JavaBean

\* @公司 青软实训

\* @作者 fengjj

\*/

public class ResumeBasicinfo {

// 简历标识

private int basicinfoId;

// 姓名

private String realName;

// 性别

private String gender;

// 出生日期

private Date birthday;

// 当前所在地

private String currentLoc;

// 户口所在地

private String residentLoc;

// 手机

private String telephone;

// 邮件

private String email;

// 求职意向

private String jobIntension;

// 工作经验

private String jobExperience;

// 头像

private String headShot;

public ResumeBasicinfo() {

}

public ResumeBasicinfo(String realName, String gender, Date birthday,

String currentLoc, String residentLoc, String telephone,

String email, String jobIntension, String jobExperience) {

super();

this.realName = realName;

this.gender = gender;

this.birthday = birthday;

this.currentLoc = currentLoc;

this.residentLoc = residentLoc;

this.telephone = telephone;

this.email = email;

this.jobIntension = jobIntension;

this.jobExperience = jobExperience;

}

......省略setter和getter方法

}

1. 在项目com.qst.itofferbacker.javabean包下创建公共分页JavaBean文件PageBean.java，用来封装分页所需的数据和操作，具体功能实现代码如下所示。

**【代码6- 5】PageBean.java**

package com.qst.itofferbacker.javabean;

import java.util.ArrayList;

import java.util.List;

/\*\*

\* 用户列表分页JavaBean

\* @公司 青软实训

\* @作者 fengjj

\*/

public class PageBean<T> {

// 每页显示记录数

private int pageSize = 10;

// 当前页码

private int pageNo = 1;

// 总页数

private int totalPages;

// 总记录数

private int recordCount;

// 每页数据记录集合

private List<T> pageData = new ArrayList<T>();

// 是否有下一页

private boolean hasNextPage;

// 是否有上一页

private boolean hasPreviousPage;

public PageBean() {

}

public PageBean(int pageSize, int pageNo) {

this.pageSize = pageSize;

this.pageNo = pageNo;

}

public int getTotalPages() {

return (recordCount + pageSize - 1) / pageSize;

}

public List<T> getPageData() {

return pageData;

}

public void setPageData(List<T> pageData) {

this.pageData = pageData;

}

public boolean isHasNextPage() {

return (this.getPageNo() < this.getTotalPages());

}

public void setHasNextPage(boolean hasNextPage) {

this.hasNextPage = hasNextPage;

}

public boolean isHasPreviousPage() {

return (this.getPageNo() > 1);

}

public void setHasPreviousPage(boolean hasPreviousPage) {

this.hasPreviousPage = hasPreviousPage;

}

......省略其它属性的setter和getter方法

}

1. 在项目manage文件夹下根据resumeList.html模板创建简历分页显示JSP页面resumeList.jsp，使用JSTL核心标签库和EL表达式从请求域中获取PageBean对象进行数据显示，具体实现代码如下所示。

在代码实现前，注意需要先在项目WebContent->WEB-INF->lib子目录下加入JSTL标签库的jar包。

**【代码6- 6】resumeList.jsp**

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

**<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>**

<%

String basePath = request.getScheme()+"://"+request.getServerName()+":"

+request.getServerPort()+request.getContextPath()+"/";

%>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>简历列表</title>

<base href="<%=basePath%>">

<link href="css/manageadmin.css" rel="stylesheet" type="text/css" />

</head>

<body>

<div class="place"> <span>位置：</span>

<ul class="placeul">

<li><a href="index.html">首页</a></li>

<li>简历列表</li>

</ul>

</div>

<div class="rightinfo">

<div class="tools">

<ul class="toolbar">

<li><span><img src="images/t03.png" /></span><a href="#">删除</a></li>

</ul>

</div>

<table class="imgtable">

<thead>

<tr>

<th ><input name="" type="checkbox" value="" checked="checked"/></th>

<th>姓名</th>

<th>手机</th>

<th>Email</th>

<th>工作经验</th>

<th>求职意向</th>

<th>操作</th>

</tr>

</thead>

<tbody>

**<c:forEach items="${requestScope.pageBean.pageData}" var="resume">**

<tr height="50px">

<td ><input name="" type="checkbox" value="" /></td>

<td>**${resume.realName}<**/td>

<td>**${resume.telephone}<**/td>

<td>**${resume.email}**</td>

<td>**${resume.jobExperience}**</td>

<td>**${resume.jobIntension}**</td>

<td ><a href="#" class="tablelink">查看</a> &nbsp;&nbsp;<a href="#" class="tablelink"> 删除</a></td>

</tr>

</c:forEach>

</tbody>

</table>

<div class="pagin">

<div class="message">共<i class="blue">

**${requestScope.pageBean.totalPages}**</i>页，

当前显示第&nbsp;<i class="blue">**${requestScope.pageBean.pageNo}**

&nbsp;</i>页</div>

<ul class="paginList">

<li class="paginItem">**<a href="ResumeServlet?type=list&pageNo=1">**

首页</a></li>

**<c:if test="${requestScope.pageBean.hasPreviousPage}">**

<li class="paginItem">

<a href="**ResumeServlet?type=list&pageNo=**

**${requestScope.pageBean.pageNo-1}**">

上一页<span class="pagepre"></span></a></li>

**</c:if>**

**<c:if test="${requestScope.pageBean.hasNextPage}">**

<li class="paginItem">

<a href="**ResumeServlet?type=list&pageNo=**

**${requestScope.pageBean.pageNo+1}**">

下一页<span class="pagenxt"></span></a></li>

</c:if>

<li class="paginItem">

<a href="**ResumeServlet?type=list&pageNo=**

**${requestScope.pageBean.totalPages}**">尾页</a></li>

</ul>

</div>

</div>

</body>

</html>



图6- 2 简历分页查询页面运行效果

## **【实验6-2】简历详情展示**

### 知识点回顾

* EL表达式中与范围有关的隐含对象
* EL表达式中pageContext隐含对象
* JSTL核心标签库

### 功能描述

实现“Q-ITOffer”锐聘网站后台管理系统的简历详情展示功能。

### 实验分析

1. 在简历分页显示页面resumeList.jsp中发起对某个简历详情进行查看的Servlet请求。
2. 在简历信息处理Servlet文件ResumeServlet.java中增加对简历详情查看请求的处理，在此请求中调用简历数据库操作类ResumeDAO获取并封装该简历的详细数据，再将封装的数据对象转发到简历详情展示页面resumeView.jsp。
3. 创建简历信息数据库操作DAO类ResumeDAO.java，实现对简历详细数据的查询和封装。
4. 创建简历详情展示JSP页面resumeView.jsp，从请求域中获取简历信息JavaBean对象进行数据输出显示。resumeView.jsp页面运行效果如图6- 4所示。
5. 上述各资源组件间的关系如图6- 3所示。



图6- 3 企业列表功能组件关系图

### 实验步骤

1. 在resumeList.jsp页面中设定“查看”功能的超链接，向ResumeServlet发送查询请求，设定请求参数为type=select，同时设定请求查看的简历标识，具体实现代码如下所示。

**【代码6- 7】resumeList.jsp**

<a href="ResumeServlet?type=select&resumeId=${resume.basicinfoId}"

class="tablelink">查看</a>

1. 在ResumeServlet.java文件中增加根据请求的简历标识进行简历详细信息查询的请求的处理，具体实现代码如下所示。

**【代码6- 8】ResumeServlet.java**

@WebServlet("/ResumeServlet")

public class ResumeServlet extends HttpServlet {

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// 设置请求字符编码

request.setCharacterEncoding("UTF-8");

String type = request.getParameter("type");

if("list".equals(type)){

......

}else **if("select".equals(type)){**

**int resumeId = Integer.parseInt(request.getParameter("resumeId"));**

**ResumeDAO dao = new ResumeDAO();**

**ResumeBasicinfo resume = dao.selectBasicinfoByID(resumeId);**

**request.setAttribute("resume", resume);**

**request.getRequestDispatcher("manage/resumeView.jsp")**

**.forward(request, response);**

**}**

}

}

1. 在简历数据库操作类ResumeDAO中增加根据请求的简历标识进行简历详细信息查询的方法selectBasicinfoID()，具体实现代码如下所示。

**【代码6- 9】ResumeDAO.java**

public class ResumeDAO {

/\*\*

\* 根据简历标识查询简历基本信息

\* @param applicantID

\* @return

\*/

public ResumeBasicinfo selectBasicinfoByID(int resumeId){

ResumeBasicinfo resume = new ResumeBasicinfo();

String sql = "SELECT \* FROM tb\_resume\_basicinfo WHERE basicinfo\_id=? ";

Connection conn = DBUtil.getConnection();

PreparedStatement pstmt = null;

try {

pstmt = conn.prepareStatement(sql);

pstmt.setInt(1, resumeId);

ResultSet rs = pstmt.executeQuery();

if(rs.next()){

resume.setBasicinfoId(rs.getInt("basicinfo\_id"));

resume.setBirthday(rs.getDate("birthday"));

resume.setCurrentLoc(rs.getString("current\_loc"));

resume.setEmail(rs.getString("email"));

resume.setGender(rs.getString("gender"));

resume.setHeadShot(rs.getString("head\_shot"));

resume.setJobExperience(rs.getString("job\_experience"));

resume.setJobIntension(rs.getString("job\_intension"));

resume.setRealName(rs.getString("realname"));

resume.setResidentLoc(rs.getString("resident\_loc"));

resume.setTelephone(rs.getString("telephone"));

}

} catch (SQLException e) {

e.printStackTrace();

} finally {

DBUtil.closeJDBC(null, pstmt, conn);

}

return resume;

}

}

1. 在manage目录下根据模板resumeView.html文件创建简历详情展示页面resumeView.jsp，页面使用EL表达式从请求域属性中获取ResumeBasicinfo对象进行简历信息的获取和显示，具体实现代码如下所示。

需要注意的是，为了使在前台项目中用户上传的简历照片在网站后台也可以查看，这里我们可以在Tomcat根目录下的upload目录中创建resumePhoto子目录，将教材Q\_ITOffer项目的applicant/images目录下求职者上传的简历头像文件拷贝到resumePhoto子目录中。

**【代码6- 10】resumeView.jsp**

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<%

String basePath = request.getScheme()+"://"+request.getServerName()+":"

+request.getServerPort()+request.getContextPath()+"/";

%>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>简历详情</title>

<base href="<%=basePath%>">

<link href="css/manageadmin.css" rel="stylesheet" type="text/css" />

</head>

<body>

<div class="place"> <span>位置：</span>

<ul class="placeul">

<li><a href="index.html">首页</a></li>

<li>简历详情</li>

</ul>

</div>

<div class="formbody">

<div class="usual">

<div class="tabson">

<ul class="forminfo">

<li>

<label>真实姓名：</label><label style="width: 200px;"><b style="color: #000">**${requestScope.resume.realName }**</b></label>

</li>

<li>

<label>照片：</label>

<img src="**/upload/resumePhoto/${requestScope.resume.headShot }**"

width="100px" height="100px">

</li>

<li>

<label>性别：</label><label style="width: 200px;">

<b style="color: #000">**${requestScope.resume.gender }<**/b></label>

</li>

<li>

<label>出生日期：</label><label style="width: 200px;">

<b style="color: #000">**${requestScope.resume.birthday }**</b></label>

</li>

<li>

<label>当前所在地：</label><label style="width: 200px;">

<b style="color: #000">

**${requestScope.resume.currentLoc }**</b></label>

</li>

<li>

<label>户口所在地：</label><label style="width: 200px;">

<b style="color: #000">

**${requestScope.resume.residentLoc }**</b></label>

</li>

<li>

<label>手机号：</label><label style="width: 200px;">

<b style="color: #000">

**${requestScope.resume.telephone }**</b></label>

</li>

<li>

<label>邮箱：</label><label style="width: 200px;">

<b style="color: #000">**${requestScope.resume.email }**</b></label>

</li>

<li>

<label>求职意向：</label><label style="width: 200px;">

<b style="color: #000">

**${requestScope.resume.jobIntension }**</b></label>

</li>

<li>

<label>工作经验：</label><label style="width: 200px;">

<b style="color: #000">

**${requestScope.resume.jobExperience }**</b></label>

</li>

</ul>

</div>

</div>

</div>

</body>

</html>



图6- 4 企业列表功能组件关系图

## **【实验6-3】依据用户角色的系统菜单设置**

### 知识点回顾

* EL表达式中与范围有关的隐含对象
* EL表达式中pageContext隐含对象
* JSTL核心标签库

### 功能描述

实现“Q-ITOffer”锐聘网站后台管理系统的通用头部页面中的登陆用户名显示功能以及主界面操作菜单依据登陆用户的不同角色进行相应的展示功能。

### 实验分析

1. 通过模板top.html文件创建系统主界面的头部JSP页面top.jsp，从而使用EL表达式获取并输出显示当前登陆的用户姓名。
2. 通过模板left.html文件创建系统主界面的操作菜单页面left.jsp，从而使用JSTL和EL依据用户的角色进行相应操作菜单的展示。
3. 修改主界面文件main.html中对新头部文件和操作菜单文件的链接地址。当一个企业管理员角色的用户成功登陆进入main.html页面后的运行效果如图6- 5所示。

### 实验步骤

1. 修改系统主界面main.html中通过<frame>标签对新的头文件和操作菜单文件的链接地址，实现代码如下所示。

**【代码6- 11】main.html**

<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>Q\_ITOffer锐聘网后台管理系统</title>

</head>

<frameset rows="88,\*" cols="\*" frameborder="no" border="0" framespacing="0">

<frame src="**top.jsp**" name="topFrame" scrolling="No" noresize="noresize" id="topFrame" title="topFrame" />

<frameset cols="187,\*" frameborder="no" border="0" framespacing="0">

<frame src="**left.jsp**" name="leftFrame" scrolling="No" noresize="noresize" id="leftFrame" title="leftFrame" />

<frame src="index.html" name="rightFrame" id="rightFrame" title="rightFrame" />

</frameset>

</frameset>

<noframes>

<body>

</body>

</noframes>

</html>

1. 在项目manage文件夹下根据模板top.html创建top.jsp页面，在显示登录用户名处使用EL表达式获取当前登陆者的用户名，实现代码如下所示。

**【代码6- 12】top.jsp**

<div class="user"><span>

${sessionScope.SESSION\_USER.userRealname}</span></div>

1. 在项目manage文件夹下根据模本left.html创建left.jsp页面，使用JSTL标签判断当前登陆用户的角色，依据角色类型进行操作菜单的显示，具体实现代码如下所示。

**【代码6- 13】left.jsp**

......

<body style="background:#f0f9fd;">

<div class="lefttop"><span></span>功能菜单</div>

<dl class="leftmenu">

**<%--仅当用户角色为管理员或企业用户时可查看 --%>**

**<c:if test="${sessionScope.SESSION\_USER.userRole == 1 ||**

**sessionScope.SESSION\_USER.userRole == 2}">**

<dd>

<div class="title"> <span><img src="../images/leftico01.png" /></span>

企业职位管理</div>

<ul class="menuson">

<li><cite></cite><a href="#" target="rightFrame">

职位申请查看</a><i></i></li>

<li><cite></cite><a href="#" target="rightFrame">职位管理</a><i></i></li>

<li><cite></cite><a href="../CompanyServlet?type=list"

target="rightFrame">企业管理</a><i></i></li>

</ul>

</dd>

**</c:if>**

**<%--所有登录用户角色均可查看 --%>**

**<c:if test="${sessionScope.SESSION\_USER.userRole == 1 ||**

**sessionScope.SESSION\_USER.userRole == 2 ||**

**sessionScope.SESSION\_USER.userRole == 3}">**

<dd>

<div class="title"> <span><img src="../images/leftico02.png" /></span>

简历管理</div>

<ul class="menuson">

<li><cite></cite><a href="../ResumeServlet?type=list" target="rightFrame"> 简历查询</a><i></i></li>

</ul>

</dd>

**</c:if>**

**<%--仅系统管理员角色可查看 --%>**

**<c:if test="${sessionScope.SESSION\_USER.userRole == 1 }">**

<dd>

<div class="title"><span><img src="../images/leftico03.png"/></span>

用户管理</div>

<ul class="menuson">

<li><cite></cite><a href="userList.jsp" target="rightFrame">

用户管理</a><i></i></li>

</ul>

</dd>

<dd>

<div class="title"><span><img src="../images/leftico04.png" /></span>

系统管理</div>

<ul class="menuson">

<li><cite></cite><a href="#" target="rightFrame">在线用户</a><i></i></li>

</ul>

</dd>

**</c:if>**

**<%--所有用户角色均可查看 --%>**

<dd>

<div class="title"><span><img src="../images/leftico04.png" /></span>

<a href="#" target="rightFrame">密码修改</a></div>

</dd>

</dl>

</body>

1. 使用一个具有企业管理员角色的用户进行登陆测试，用户登陆成功后，主界面main.html的运行效果如图6- 5所示。



图6- 5 系统主界面运行效果图