软件测试报告

目录

一 测试需求分析

1 系统概述

随着企业内人力资源管理的网络化和系统化越来越完善，并且越来越科学化。人力资源系统在企业管理中也备受企业管理者的青睐。人力资源管理系统包括人事日常事务、工资、培训、人事资料等管理。是一个为制定人力资源决策提供信息的集成系统，是为了提高系统管理者人力资源管理水平而开发的。主要的目标就是能够让企业管理者方便快捷地掌握员工的个人信息，工作进度和工作状态等，快速正确地进行决策。降低企业人力资源管理的人力以及成本，提高人力资源管理的效率。

人力资源管理系统在企业的有效实施，会促进企业人力资源管理向规范化、标准化、决策科学化发展;促使企业管理者能缓解工作量巨大的压力，避免以前工作中出现的错误，减少出错的几率;能够让企业管理者专注于对企业的人力资源管理活动进行计划，组织，监督和咨询职能；并且对企业的人力资源管理各个方面进行认真地分析、详细地规划、准确地实施、调整。调动所有的有利因素，来提高企业人力资源管理水平和企业管理者的效率，最终使人成为企业经营发展中真正的第一资源。

系统特点：操作简单、界面清晰友好，功能强大、运行稳定快速、系统资源占用少。

2 测试需求

本次测试针对开发的员工管理系统进行，包括功能测试，界面测试，员工管理测试，信息查询测试，增加员工测试，删除员工测试，管理测试。按照规格需求说明书中的功能进行测试，在测试过程中发现软件的漏洞不足并予以改正。

3 功能模块

3.1 员工管理模块功能需求

管理员可以添加员工的信息，并可以对添加的用户信息进行查询、修改、删除。

3.2 部门管理模块功能需求

管理员可以添加部门的信息，并可以对添加的部门信息进行修改、查询、删除。

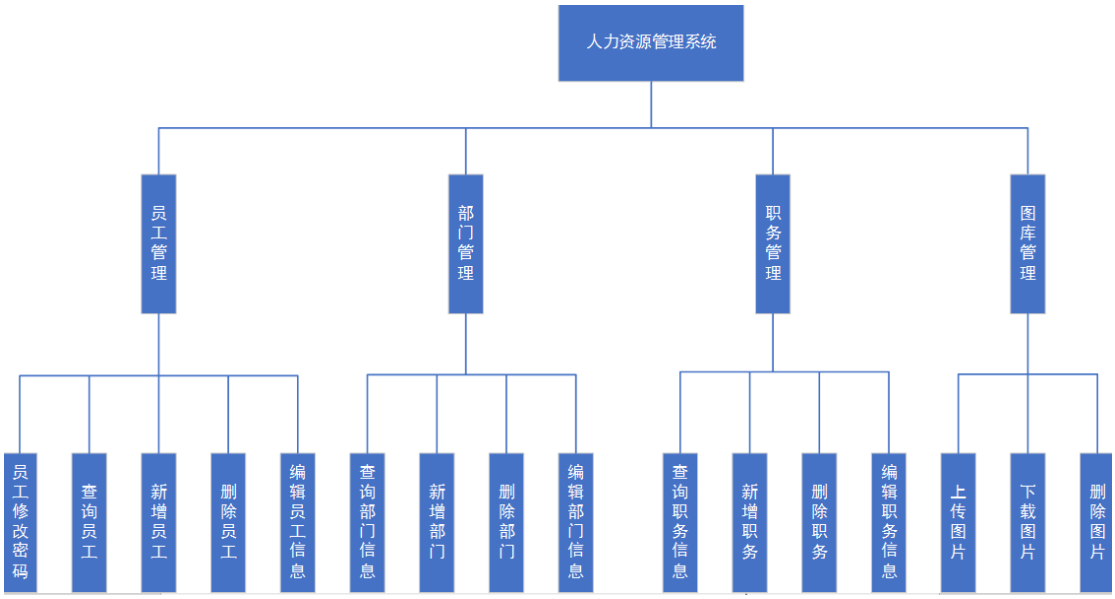
3.3 职务管理

管理员可以添加职务的信息，并可以对添加的职务信息进行修改、查询、删除。

3.4 图库管理

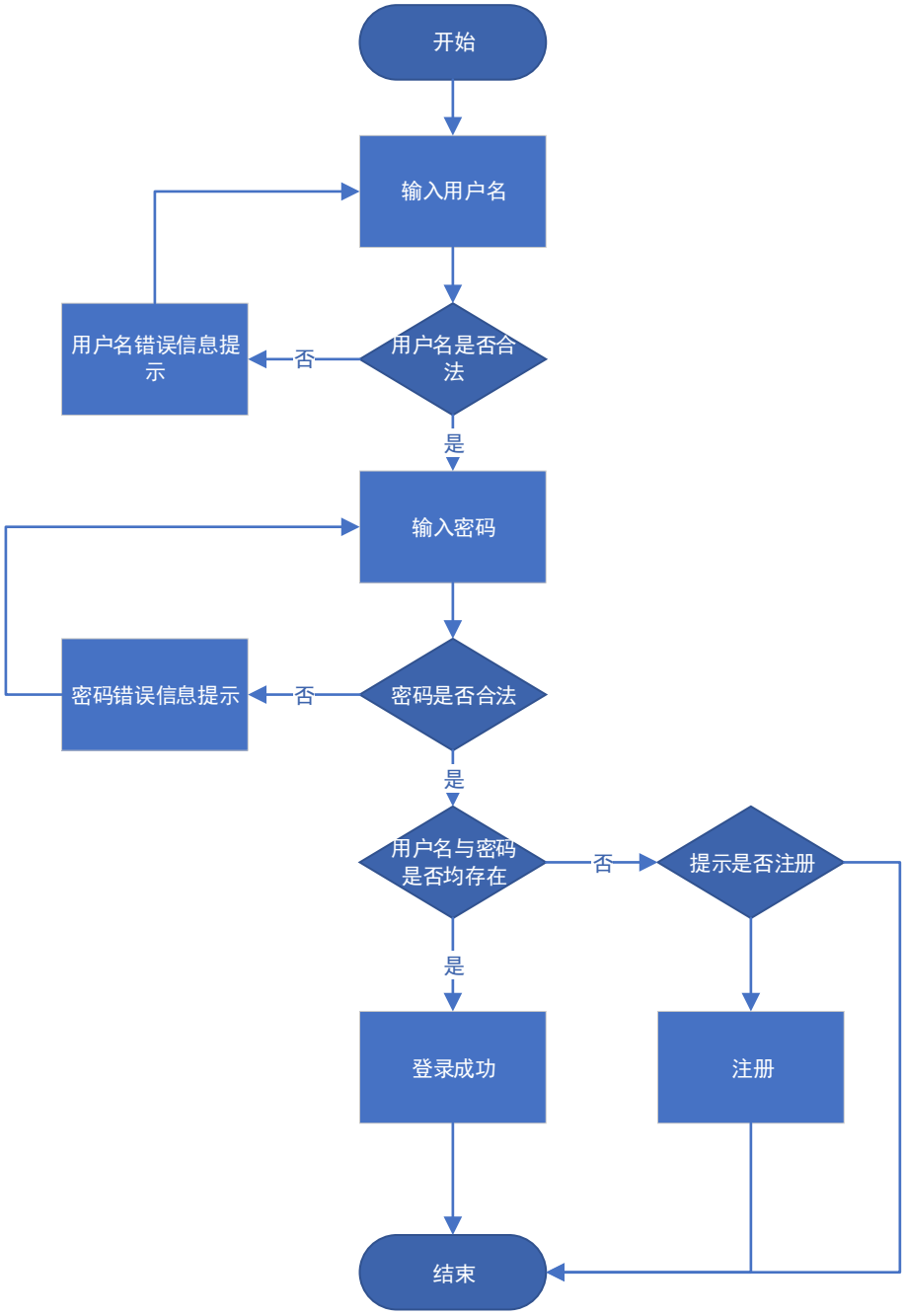
管理员可以添加图库的信息，并可以对添加的职务信息进行修改、查询、删除。

人力资源管理系统分为管理员和普通员工两个模块，此次测试模块为管理员管理员工子模块，具体包括员工管理模块，部门管理模块，职务管理模块，图库管理模块。实现了各个子模块的增删改查等基本操作，还增加了数据分析，分页查询等功能。

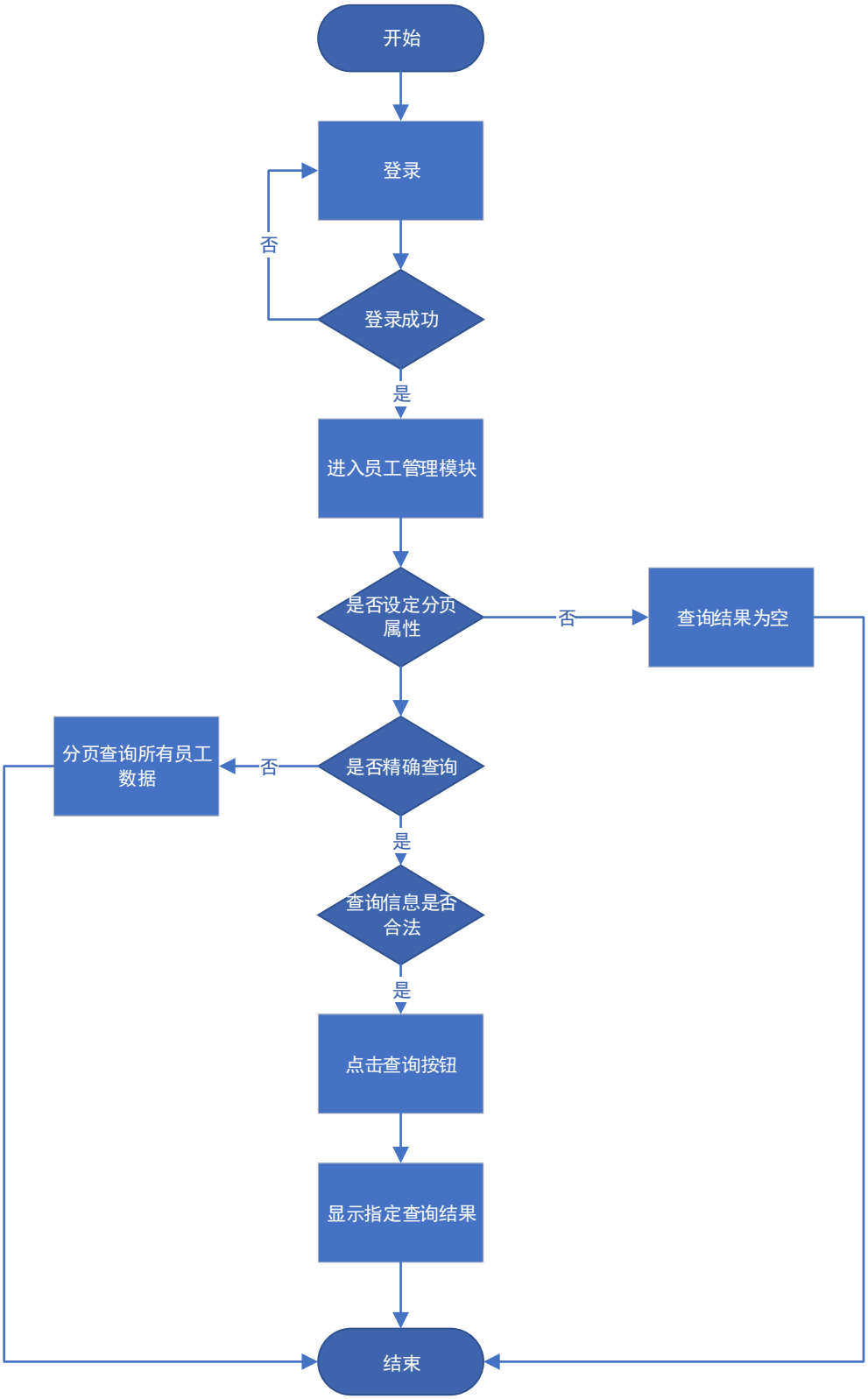


4 详细设计

4.1 管理员登录系统流程图



4.2 管理员分页查询员工信息流程图



二 制定测试计划

1 测试目的

1.1 练习和掌握软件测试管理的一般过程与步骤；

1.2 掌握测试管理的人工过程和能够通过相关管理软件实现以下工作：

1) 配置软件资产信息、软件需求、软件模型和缺陷数据库；

2) 创建和管理多个测试组和用户；

3) 配置测试环境、编写详细测试计划、安排测试进度；

4) 设计测试脚本、测试用例；

5) 实施测试、执行测试和评估测试。

2 测试内容

选题一：管理员登录测试；

选题二：对管理员员工管理分页查询功能模块的单元测试。

选题三：对整个系统做自动化测试

3 测试人员及任务

测试人员：赵雷

测试任务：

1) 软件测试计划及相关资料的编写与收集

2) 对特定问题编写程序代码并对其进行黑盒测试

3) 对特定问题编写程序代码，并对其进行白盒测试

4) 使用 JMeter 以及 Badboy 进行自动化测试

5) 利用 SQLMAP 进行 SQL 注入检测

4 测试方法

对选题一即管理员登录选用黑盒测试的方法，利用黑盒测试中的边界值测试，等价类划分测试以及决策表测试。

对选题二即管理员员工管理模块功能的进行白盒测试，白盒测试代码为员工管理 DAOImpl(Data Access Object Implements，数据访问对象实现层)编写白盒测试类，利用 JUnit4 进行单元测试，计算覆盖率。具体采用了语句覆盖，条件覆盖，分支覆盖等三种白盒测试方法。

对选题一与选题二做 SQL 注入检测。

对选题三即整个项目进行自动化测试，具体包括压力测试，性能测试等。使用 Badboy 录制测试脚本，利用 JMeter 进行分析。

三 测试用例设计

1 用户登录黑盒测试用例设计

1.1 边界值测试设计

人们从长期的测试工作经验得知，大量的错误是发生在输入或输出范围的边界上的，而不是在输入范围的内部。因此针对各种边界情况设计测试用例，可以查出更多的错误。

在管理员登录测试用例中，登陆窗口的账号和密码应该有一定的限制和规范，在注册页面发现注册用户名长度为固定值 8 位，密码长度未做限制。

用户名长度只可为 8 位

密码长度设置在 6-20 位之间

选定登录用户名:15427042 密码:123456

表 1 健壮性边界值测试

用例编号	用户名	密码	预期输出	实际输出
1	空	123456	用户名不能为空	账号错误
2	1	123456	用户名长度为 8 位	账号错误
3	1234567	123456	用户名长度为 8 位	账号错误
4	12345678	123456	用户名不存在	账号错误
5	123456789	123456	用户名长度为 8 位	账号错误
6	15427042	123456	登录成功	登录成功
7	15427042	空	密码不能为空	密码错误
8	15427042	1	密码长度低于 6 位	密码错误
9	15427042	12345	密码长度低于 6	密码错误

			位	
10	15427042	123456789101112	密码错误	密码错误
11	15427042	1234567123456711111	密码错误	密码错误
12	1M 字符	12345123451234512345	用户名长度为 8 位	账号错误
13	15427042	123451234512345123451	密码长度超过 20 位	密码错误

1.2 等价类划分设计

等价类划分是一种典型的黑盒测试方法，用这一方法设计测试用例完全不考虑程序的内部结构，只根据对程序的要求和说明，即需求规格说明书。我们必须仔细分析和推敲说明书的各项需求，特别是功能需求。把说明中对输入的要求和输出的要求区别开来并加以分解。

等价类划分的办法是把程序的输入域划分成若干部分，然后从每个部分中选取少数代表性数据作为测试用例。每一类的代表性数据在测试中的作用等价于这一类中的其他值，也就是说，如果某一类中的一个例子发现了错误，这一等价类中的其他例子也能发现同样的错误；反之，如果某一类中的一个例子没有发现错误，则这一类中的其他例子也不会查出错误（除非等价类中的某些例子属于另一等价类，因为几个等价类是可能相交的）。使用这一方法设计测试用例，首先必须在分析需求规格说明的基础上划分等价类，列出等价类表。

表 2 登录窗口等价类划分描述

输入条件	输入用户名与密码	有效等价类	无效等价类
		1.字符串 2.用户名与密码 3.用户名与密码均无空格等特殊字符	8.用户名含空格等特殊字符 9.密码中含空格等特殊字符 10.用户名与密码均含有空格等特殊字符
输出条件	用户名错误	4.用户名长度为 8 且密码长度介于	11.用户名为空 12.密码为空

		6-20	13.用户名与密码均为空 14.用户名长度小于8位，密码合法 15 用户名长度大于8位，密码合法 16.密码长度小于6位，用户名合法 17.密码长度大于20位，用户名合法 18.用户名与密码均不合法
	密码错误	5.用户名正确且密码错误	19.用户名不正确但密码正确 20.用户名与密码均不正确
	登录成功	6.用户名与密码均正确	21.用户名与密码至少一个不正确
	用户不存在	7.用户名与密码均合法但数据库中无此记录	

表 3 用户登录等价类划分测试用例

用例编号	包含等价类	用户名	密码	预期输出	实际输出
1	1,2,3,11,14,19	空	123456	用户名不能为空	账号错误
2	1,2,3,14,19	1	123456	用户名长度为8位	账号错误

3	1,2,3,14,19,21	1234	123456	用户名长度为 8 位	账号错误
4	1,2,3,15,19,21	1234567	123456	用户名长度为 8 位	账号错误
5	1,2,3,15,19,21	12345678	123456	用户名不存在	账号错误
6	1,2,3,15,19,21	123456789	123456	用户名长度为 8 位	账号错误
7	1,2,3,4,6,21	15427042	123456	登录成功	登录成功
8	1,2,3,4,12,21	15427042	空	密码不能为空	密码错误
9	1,2,3,4,21	15427042	1	密码长度低于 6 位	密码错误
10	1,2,3,4,21	15427042	12345	密码长度低于 6 位	密码错误
11	1,2,3,4,21	15427042	123456789101112	密码错误	密码错误
12	1,2,3,4,21	15427042	1234567123456711111	密码错误	密码错误
13	1,2,3,4,18,20,21	1M 字符	12345123451234512345	用户名长度为 8 位	账号错误
14	1,2,3,4,21	15427042	123451234512345123451	密码长度超过 20 位	密码错误
15	1,2,3,4,8,19,21	154 2	123456	用户名含有特殊字符	密码错误
16	1,2,3,4,9,20,21	15427042	123 6	密码错误	密码错误
17	1,2,3,4,9,20,21	15270 23	1234 6	用户名含有特殊字符	账号错误
18	1,2,3,4,8,19,21	15427.42	123456	用户名含有特殊字符	账号错误
19	1,2,3,4,8,19,21	15427042	1234.6	密码错误	密码错误
20	1,2,3,4,8,9,20,21	15470..2	123..6	用户名含有特殊字符	账号错误
21	1,2,3,4,8,19,21	管理员	123456	用户名不能含有汉字	账号错误
22	1,2,3,4,8,20,21	管理员	12345678	用户名不能含	账号错误

				有汉字	

1.3 决策表

决策表又称判断表，是一种呈表格状的图形工具，适用于描述处理判断条件较多，各条件又相互组合、有多种决策方案的情况。精确而简洁描述复杂逻辑的方式，将多个条件与这些条件满足后要执行动作相对应。但不同于传统程序语言中的控制语句，决策表能将多个独立的条件和多个动作直接的联系清晰的表示出来。

在用户登录测试中，含有用户名长度为 8 位，密码长度位 6-20 位以及用户名和密码位字符串三个条件，每个条件有两种取值，故共含有 2*2*2=8 种规则。

表 4 用户登录决策表

		1	2	3	4	5	6	7	8
条件	C1 用户名是字符串吗?	N	N	Y	Y	Y	Y	N	Y
	C2 用户名长度是 8 位吗?	N	Y	N	Y	Y	N	N	Y
	C3 密码长度在 6-20 之间吗?	Y	N	Y	Y	N	N	N	Y
动作	A1 用户名错误	X	X	X				X	
	A2 密码错误					X	X		
	A3 登录成功				X				
	A4 查无此人								X

表 5 用户登录决策表测试用例

用例编号	用户名	密码	预期输出	实际输出
1	!@#$%^&*	123456	用户名含有特殊字符	账号错误
2	********	123456	用户名含有特殊字符	账号错误
3	15427042	123456	登录成功	登录成功
4	1122232432432543	12345	用户名错误	账号错误
5	15427042	123 6	密码错误	密码错误
6	15427042	12345	密码错误	密码错误
7	!@#$%^&*	11	用户名含有特殊字符	账号错误
8	15427099	123456	查无此人	账号错误

2 员工分页查询白盒测试用例设计

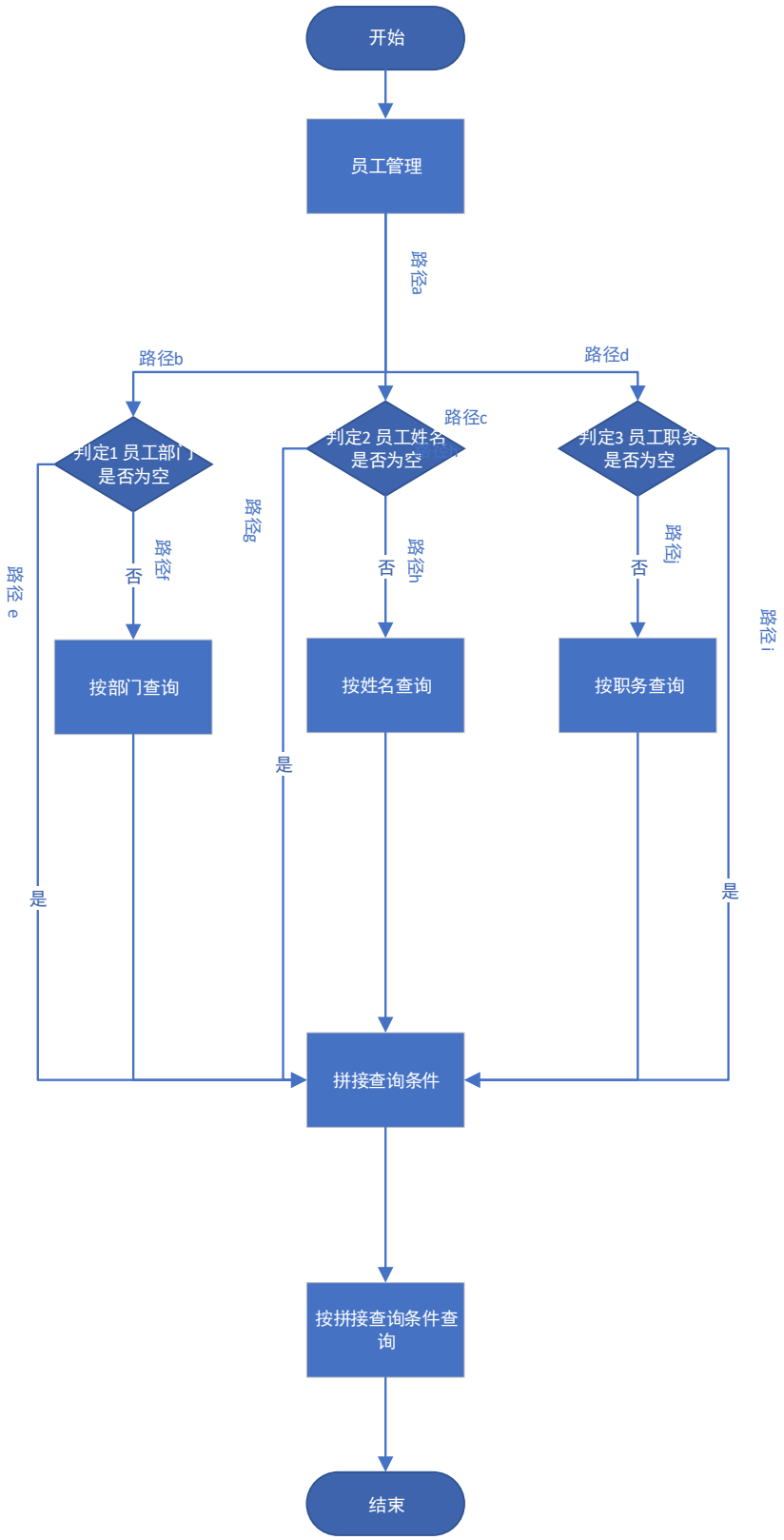
2.1 被测代码

```
@Override
    /**
     * 分页查询
     */
    public Pager<Emp> queryByPage(Emp condition, int pageNum, int
pageSize) throws SQLException {
        Pager<Emp> pager = new Pager<Emp>();
        ArrayList<String> params = new ArrayList<String>();

        String empName = condition.getEmpName();
        String empDeptName = condition.getDept().getDeptName();
        String empJobName = condition.getJob().getJobName();

        StringBuilder sql = new StringBuilder("select * from emp join
dept on(emp.emp_dept_id=dept.dept_id) join job
on(emp.emp_job_id=job.job_id) where 1=1");
        StringBuilder countSql = new StringBuilder("select count(*) from
emp join dept on(emp.emp_dept_id=dept.dept_id) join job
on(emp.emp_job_id=job.job_id) where 1=1");

        if (StringUtil.isNotEmpty(empName)) {
            sql.append(" and emp_name = ? ");
            countSql.append(" and emp_name = ? ");
            params.add(empName);
        }

        if (StringUtil.isNotEmpty(empDeptName)) {
            sql.append(" and dept_name = ? ");
            countSql.append(" and dept_name = ? ");
            params.add(empDeptName);
```

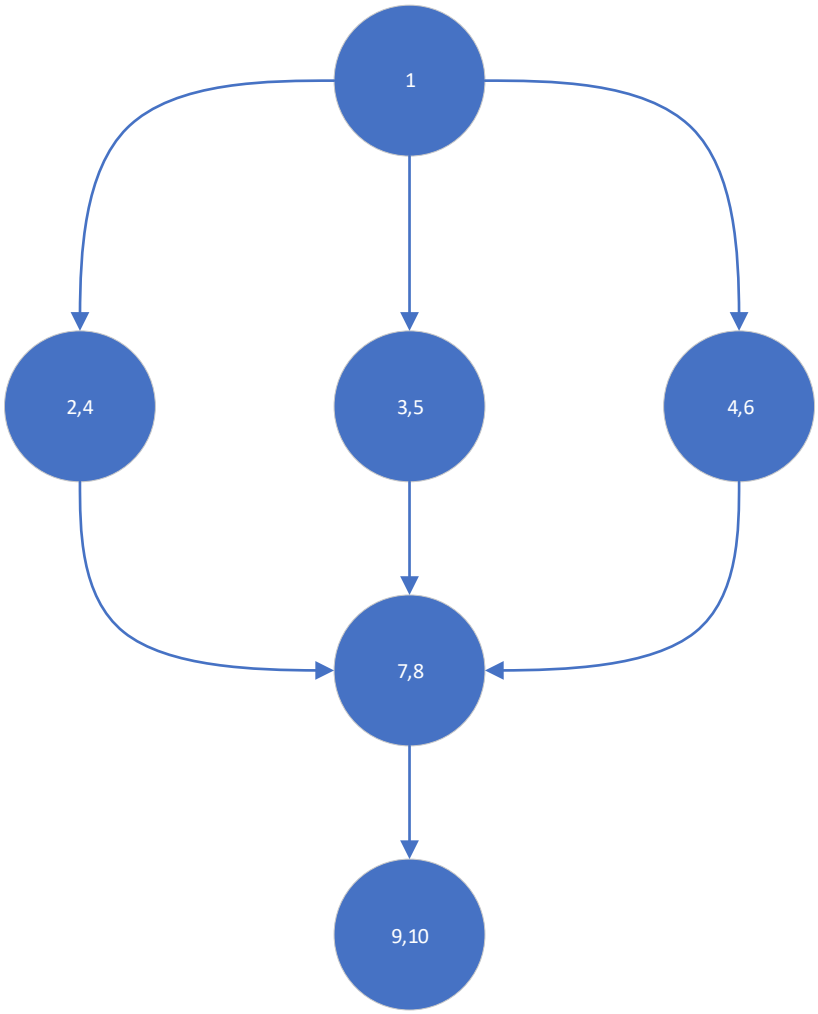
```
        }

        if (StringUtil.isNotEmpty(empJobName)) {
            sql.append(" and job_name = ? ");
            countSql.append(" and job_name = ? ");
            params.add(empJobName);
        }

        int fromIndex = pageSize * (pageNum - 1);
        int toIndex = pageSize * pageNum;
        ArrayList<Emp> empList = new ArrayList<Emp>();
        Connection conn = DbUtil.getConnection();
        String pageSql = pager.getSql(sql.toString(), fromIndex, toIndex);
        PreparedStatement pstmt = null;
        pstmt = conn.prepareStatement(pageSql);
        for (int i = 0; i < params.size(); i++) {
            pstmt.setObject(i + 1, params.get(i));
        }

        Emp emp = null;
        DeptDao ddao = DAOFactory.instance().getDeptDao();
        JobDao jdao = DAOFactory.instance().getJobDao();
        ResultSet rs = null;
        rs = pstmt.executeQuery();
        while (rs.next()) {
            emp = new Emp(rs.getInt(1), rs.getString(2), rs.getString(3),
rs.getString(4), rs.getString(5),
                    rs.getString(6), rs.getInt(7),
ddao.queryById(rs.getInt(8)), jdao.queryById(rs.getInt(9)),
                    rs.getString(10));
            empList.add(emp);
        }
        pstmt = conn.prepareStatement(countSql.toString());
        for (int i = 0; i < params.size(); i++) {
```

```
                pstmt.setObject(i + 1, params.get(i));
            }
            rs = pstmt.executeQuery();
            int totalRecord = 0;
            if (rs.next()) {
                totalRecord = rs.getInt(1);
            }

            int totalPage = totalRecord / pageSize;
            if (totalRecord % pageSize != 0) {
                totalPage = totalPage + 1;
            }
            pager = new Pager<Emp>(pageSize, pageNum, totalRecord,
totalPage, empList);
            DbUtil.close(rs, pstmt, conn);
            return pager;
        }
```

2.2 流程图


开始
员工管理
路径a
路径b
路径d
判定1 员工部门是否为空
判定2 员工姓名是否为空
判定3 员工职务是否为空
路径c
路径f
否
路径g
路径h
否
路径j
否
路径 e
是
是
路径 i
是
按部门查询
按姓名查询
按职务查询
拼接查询条件
按拼接查询条件查询
结束

2.3 程序流图



2.4 语句覆盖设计

使程序中的每一条可执行语句至少执行一次。

表 6 语句覆盖测试用例

用例编号	员工姓名	员工部门	员工职务	预期输出	实际输出	覆盖路径
1	空	空	空	全部数据分页查询结果	全部数据分页查询结果	a ,b,e;a,c,g;a,d,i
2	Trump	国家元首	经理	查询到指定数据	查询到指定数据	a,b,f;a,c,h;a,d,j

2.5 判定覆盖设计

使程序中的每个判断的取真分支和取假分支至少经历一次，即判断的真假值均被满足。判定中设计员工姓名，员工部门，员工职务三种判定条件，每种条件有真假两种取值，所以共有 2*2*2=8 种判定覆盖的结果。

表 7 判定覆盖测试用例						
用例编号	员工姓名	员工部门	员工职务	预期输出	实际输出	判定路径
1	空	空	空	全部查询数据	全部查询数据	a,b,e;a,c,g;a,d,i
2	Trump	空	空	同名员工数据	同名员工数据	a,b,f;a,c,g;a,d,i
3	空	开发部	空	同部门员工数据	同部门员工数据	a,b,e;a,c,h;a,d,i
4	空	空	经理	同职位员工数据	同职位员工数据	a,b,e;a,c,g;a,d,j
5	Trump	国家元首	空	同部门同名员工数据	同部门同名员工数据	a,b,f;a,c,h;a,d,i
6	Trump	空	经理	同职务同名员工数据	同职务同名员工数据	a,b,f;a,c,g;a,d,j
7	空	国家元首	经理	同部门同职务员工数据	同部门同职务员工数据	a,b,e;a,c,h;a,d,j
8	Trump	国家元首	经理	同部门同职务同名员工数据	同部门同职务同名员工数据	a,b,f;a,c,h;a,d,j

2.6 条件覆盖设计

使程序的判定中每个条件的真假取值至少满足一次，现设员工姓名事件为 A，员工部门事件为 B，员工职务事件为 C，则真假值分别为

AT：A 真 AF：A 假

BT：B 真 BF：B 假

CT：C 真 CF：C 假

表 8 条件覆盖测试用例

用例编号	员工姓名	员工部门	员工职务	预期输出	实际输出	覆盖条件
1	空	空	空	全部查询数据	全部查询数据	AF,BF,CF
2	Trump	空	空	同名员工数据	同名员工数据	AT,BF,CF
3	空	开发部	空	同部门员工数据	同部门员工数据	AF,BT,CF
4	空	空	经理	同职位员工数据	同职位员工数据	AF,BF,CT
5	Trump	国家元首	空	同部门同名员工数据	同部门同名员工数据	AT,BT,CF
6	Trump	空	经理	同职务同名员工数据	同职务同名员工数据	AT,BF,CT
7	空	国家元首	经理	同部门同职务员工数据	同部门同职务员工数据	AF,BT,CT
8	Trump	国家元首	经理	同部门同职务同名员工数据	同部门同职务同名员	AT,BT,CT

					工数据	

3 Junit 单元测试与覆盖率

3.1 单元测试脚本

```
package edu.dlu.hr.test;

import java.sql.SQLException;
import java.util.ArrayList;

import org.junit.Test;

import edu.dlu.hr.common.DAOFactory;
import edu.dlu.hr.common.Pager;
import edu.dlu.hr.dept.pojo.Dept;
import edu.dlu.hr.emp.dao.EmpDao;
import edu.dlu.hr.emp.pojo.Emp;
import edu.dlu.hr.job.pojo.Job;

public class TestQueryEmpByPage {
    private EmpDao dao = DAOFactory.instance().getEmpDao();
    Dept dept = null;
    Job job = null;
    Emp emp = null;
    Pager<Emp> pager = null;
    ArrayList<Emp> list = null;

    /**
     * 语句覆盖测试
     *
     * @throws SQLException
     */
    @Test
```

```
    public void testQueryByPagerStmtCover() throws SQLException {
        dept = new Dept();
        job = new Job();
        emp = new Emp(null, dept, job);
        pager = dao.queryByPage(emp, 1, 5);
        list = (ArrayList<Emp>) pager.getDataList();
        for (Emp e : list) {
            System.out.println(e);
        }
        System.out.println("-----------语句覆盖分隔符---------");
        dept = new Dept("董事会");
        job = new Job("经理");
        emp = new Emp("Trump", dept, job);
        pager = dao.queryByPage(emp, 1, 5);
        list = (ArrayList<Emp>) pager.getDataList();
        for (Emp e : list) {
            System.out.println(e);
        }
    }

    /**
     * 判定覆盖测试
     *
     * @throws SQLException
     */
    @Test
    public void testQueryByPagerDesicionCover() throws SQLException
{
        dept = new Dept();
        job = new Job();
        emp = new Emp(null, dept, job);
        pager = dao.queryByPage(emp, 1, 5);
        list = (ArrayList<Emp>) pager.getDataList();
        for (Emp e : list) {
```

```
        System.out.println(e);
    }
    System.out.println("-----------条件覆盖分隔符---------");
    dept = new Dept();
    job = new Job();
    emp = new Emp("Trump", dept, job);
    pager = dao.queryByPage(emp, 1, 5);
    list = (ArrayList<Emp>) pager.getDataList();
    for (Emp e : list) {
        System.out.println(e);
    }
    System.out.println("-----------条件覆盖分隔符---------");
    dept = new Dept("开发部");
    job = new Job();
    emp = new Emp(null, dept, job);
    pager = dao.queryByPage(emp, 1, 5);
    list = (ArrayList<Emp>) pager.getDataList();
    for (Emp e : list) {
        System.out.println(e);
    }
    System.out.println("-----------条件覆盖分隔符---------");
    dept = new Dept();
    job = new Job("经理");
    emp = new Emp(null, dept, job);
    pager = dao.queryByPage(emp, 1, 5);
    list = (ArrayList<Emp>) pager.getDataList();
    for (Emp e : list) {
        System.out.println(e);
    }
    System.out.println("-----------条件覆盖分隔符---------");
    dept = new Dept("国家元首");
    job = new Job();
    emp = new Emp("Trump", dept, job);
    pager = dao.queryByPage(emp, 1, 5);
```

```
        list = (ArrayList<Emp>) pager.getDataList();
        for (Emp e : list) {
            System.out.println(e);
        }
        System.out.println("-----------条件覆盖分隔符---------");
        dept = new Dept();
        job = new Job("经理");
        emp = new Emp(null, dept, job);
        pager = dao.queryByPage(emp, 1, 5);
        list = (ArrayList<Emp>) pager.getDataList();
        for (Emp e : list) {
            System.out.println(e);
        }
        System.out.println("-----------条件覆盖分隔符---------");
        dept = new Dept("国家元首");
        job = new Job("经理");
        emp = new Emp(null, dept, job);
        pager = dao.queryByPage(emp, 1, 5);
        list = (ArrayList<Emp>) pager.getDataList();
        for (Emp e : list) {
            System.out.println(e);
        }
        System.out.println("-----------条件覆盖分隔符---------");
        dept = new Dept("董事会");
        job = new Job("经理");
        emp = new Emp("Stefen", dept, job);
        pager = dao.queryByPage(emp, 1, 5);
        list = (ArrayList<Emp>) pager.getDataList();
        for (Emp e : list) {
            System.out.println(e);
        }
    }

    /**
```

```
     * 条件覆盖测试
     *
     * @throws SQLException
     */
    @Test
    public void testQueryByPagerConditionCover() throws
SQLException {
        dept = new Dept();
        job = new Job();
        emp = new Emp(null, dept, job);
        pager = dao.queryByPage(emp, 1, 5);
        list = (ArrayList<Emp>) pager.getDataList();
        for (Emp e : list) {
            System.out.println(e);
        }
        System.out.println("-----------条件覆盖分隔符---------");
        dept = new Dept();
        job = new Job();
        emp = new Emp("Trump", dept, job);
        pager = dao.queryByPage(emp, 1, 5);
        list = (ArrayList<Emp>) pager.getDataList();
        for (Emp e : list) {
            System.out.println(e);
        }
        System.out.println("-----------条件覆盖分隔符---------");
        dept = new Dept("开发部");
        job = new Job();
        emp = new Emp(null, dept, job);
        pager = dao.queryByPage(emp, 1, 5);
        list = (ArrayList<Emp>) pager.getDataList();
        for (Emp e : list) {
            System.out.println(e);
        }
        System.out.println("-----------条件覆盖分隔符---------");
```

```
dept = new Dept();
job = new Job("经理");
emp = new Emp(null, dept, job);
pager = dao.queryByPage(emp, 1, 5);
list = (ArrayList<Emp>) pager.getDataList();
for (Emp e : list) {
    System.out.println(e);
}
System.out.println("-----------条件覆盖分隔符---------");
dept = new Dept("国家元首");
job = new Job();
emp = new Emp("Trump", dept, job);
pager = dao.queryByPage(emp, 1, 5);
list = (ArrayList<Emp>) pager.getDataList();
for (Emp e : list) {
    System.out.println(e);
}
System.out.println("-----------条件覆盖分隔符---------");
dept = new Dept();
job = new Job("经理");
emp = new Emp(null, dept, job);
pager = dao.queryByPage(emp, 1, 5);
list = (ArrayList<Emp>) pager.getDataList();
for (Emp e : list) {
    System.out.println(e);
}
System.out.println("-----------条件覆盖分隔符---------");
dept = new Dept("国家元首");
job = new Job("经理");
emp = new Emp(null, dept, job);
pager = dao.queryByPage(emp, 1, 5);
list = (ArrayList<Emp>) pager.getDataList();
for (Emp e : list) {
    System.out.println(e);
```

```
        }
        System.out.println("-----------条件覆盖分隔符---------");
        dept = new Dept("董事会");
        job = new Job("经理");
        emp = new Emp("Stefen", dept, job);
        pager = dao.queryByPage(emp, 1, 5);
        list = (ArrayList<Emp>) pager.getDataList();
        for (Emp e : list) {
            System.out.println(e);
        }
    }
}
```

3.2 覆盖率

利用 JUnit 所自带的 Coverage As 功能进行语句覆盖，判定覆盖与条件覆盖覆盖率的统计。

结果见第四部分 单元测试结果 部分。

4 系统自动化测试

软件测试自动化的研究领域主要集中在软件测试流程的自动化管理以及动态测试的自动化（如单元测试、功能测试以及性能测试方面）。在这两个领域，与手工测试相比，测试自动化的优势是明显的。首先自动化测试可以提高测试效率，使测试人员更加专注于新的测试模块的建立和开发，从而提高测试覆盖率；其次，自动化测试更便于测试资产的数字化管理，使得测试资产在整个测试生命周期内可以得到复用，这个特点在功能测试和回归测试中尤其具有意义。

4.1 测试工具

Badboy Community, Apache-JMeter-4.0.

4.2 测试脚本

测试脚本使用 Badboy 进行录制，在录制过程中访问了所有的页面并且测试了所有提供的功能，录制结束后导出为名为 Script.jmx 的脚本文件，但是因脚本过长不粘贴。

4.3 具体执行步骤

1)利用 Badboy 自动录制测试脚本
2)导出脚本文件位 JMeter 格式，命名为 Script.jmx
3)使用 JMeter 打开 Script.jmx
4)添加监听器 View Result Tree(查看结果树),Aggregate Report(聚合报告),Summary Report(总结报告),Aggregate Graph(剧集图表),View Results in Table(查看表格结果),Graph Results(图表结果)
5)修改线程组参数,修改线程组参数(Numbers of Threads(Users))模拟 10 个用户同时进行访问,Loop Count 10,10 个用户每个用户发出 10 次请求
6)运行并查看结果

5 SQL 注入

SQL 注入攻击指的是通过构建特殊的输入作为参数传入 Web 应用程序，而这些输入大都是 SQL 语法里的一些组合，通过执行 SQL 语句进而执行攻击者所要的操作，其主要原因是程序没有细致地过滤用户输入的数据，致使非法数据侵入系统。

根据相关技术原理，SQL 注入可以分为平台层注入和代码层注入。前者由不安全的数据库配置或数据库平台的漏洞所致;后者主要是由于程序员对输入未进行细致地过滤，从而执行了非法的数据查询。基于此，SQL 注入的产生原因通常表现在以下几方面：①不当的类型处理；②不安全的数据库配置；③不合理的查询集处理；④不当的错误处理；⑤转义字符处理不合适；⑥多个提交处理不当。

5.1 测试工具

SQLMAP
Python2.7.14
DVMA

5.2 测试脚本

1)利用登陆用户名检测 SQL 注入
sqlmap.py -u “localhost:7777/hr/login.html?empLoginName=1” --batch --banner
2)利用登陆密码检测 SQL 注入
sqlmap.py -u “localhost:7777/hr/login.html?empPwd=1” --batch --banner
3)在分页查询中利用员工姓名检测 SQL 注入
sqlmap.py -u “localhost:7777/hr/QueryEmpByPage.html?empName=1” --batch –

-banner

4)在分页查询中利用员工部门检测 SQL 注入

sqlmap.py -u “localhost:7777/hr/QueryEmpByPage.html?empDeptName=1” --batch –-banner

5)在分页查询中利用员工职务检测 SQL 注入

sqlmap.py -u “localhost:7777/hr/QueryEmpByPage.html?empJobName=1” –-batch –-banner

5.3 具体执行步骤

1)配置好 Python2.7 环境变量

2)启动 Tomcat 服务器

3)进入 SQLMAP 目录,在 cmd 中使用命令行依次执行测试脚本

四 测试用例结果

1 黑盒测试结果

1.1 边界值用例测试结果

表 1 用例 1

输入账号：空 输入密码：123456

预期输出：用户名不能为空 实际输出：账号错误

表 1 用例 2

输入账号：1 输入密码：123456

预期输出：用户名长度为 8 位 实际输出：

表 1 用例 3

输入账号：1234567 输入密码：123456

预期输出：用户名长度为 8 位 实际输出

表 1 用例 4

输入账号：12345678 输入密码： 123456

预期输出：用户名不存在 实际输出

表 1 用例 5

输入账号：123456789 输入密码： 123456

预期输出：用户名长度为 8 位 实际输出

表 1 用例 6

输入账号：15427042 输入密码： 123456

预期输出：登录成功 实际输出

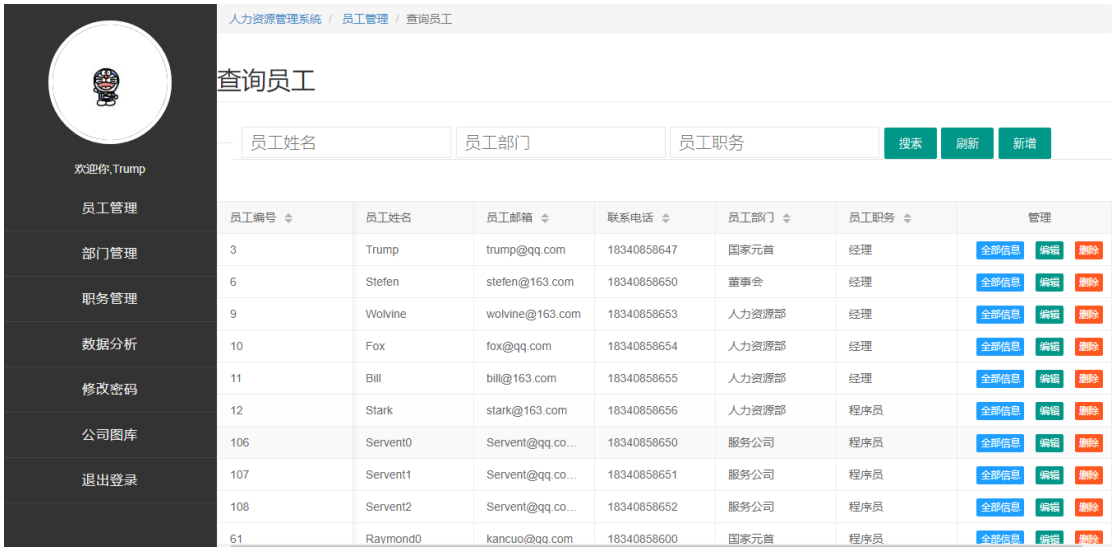


表 1 用例 7

输入账号：15427042 输入密码：空

预期输出：密码不能为空 实际输出

表 1 用例 8

输入账号：15427042 输入密码：1

预期输出：密码长度低于 6 位 实际输出

表 1 用例 9

输入账号：15427042 输入密码： 12345

预期输出：密码长度低于 6 位 实际输出

表 1 用例 10

输入账号：15427042 输入密码： 123456789101112

预期输出：密码错误 实际输出

表 1 用例 11

输入账号：15427042 输入密码： 123451234512345123451

预期输出：密码长度超过 20 实际输出

表 1 用例 12

输入账号：未测试 输入密码： 123456

预期输出：用户名超过最大字长 实际输出

表 1 用例 13

输入账号： 1542704222222222222222222222 输入密码： 22222222222222222

预期输出：账号错误 实际输出

1.2 等价类划分用例测试结果

表 3 用例 1

输入账号：空 输入密码：123456

预期输出：用户名不能为空 实际输出：账号错误

表 3 用例 2

输入账号：1 输入密码：123456

预期输出：用户名长度为 8 位 实际输出：

表 3 用例 3

输入账号：1234 输入密码：123456

预期输出：用户名长度为 8 位 实际输出

表 3 用例 4

输入账号：1234567 输入密码：123456

预期输出：用户名长度为 8 位 实际输出

表 3 用例 5

输入账号：12345678 输入密码： 123456

预期输出：用户名不存在 实际输出

表 3 用例 6

输入账号：123456789 输入密码： 123456

预期输出：用户名长度为 8 位 实际输出

表 3 用例 7

输入账号：15427042 输入密码： 123456

预期输出：登录成功 实际输出

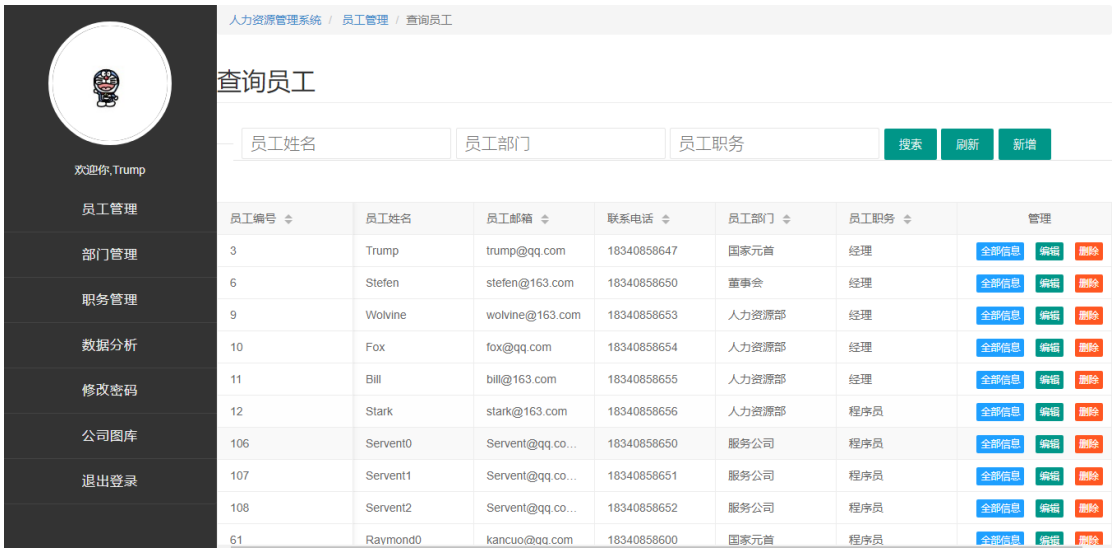


表 3 用例 8

输入账号：15427042 输入密码：空

预期输出：密码不能为空 实际输出

表 3 用例 9

输入账号：15427042 输入密码：1

预期输出：密码长度低于 6 位 实际输出

表 3 用例 10

输入账号：15427042 输入密码： 12345

预期输出：密码长度低于 6 位 实际输出

表 3 用例 11

输入账号：15427042 输入密码： 123456789101112

预期输出：密码错误 实际输出

表 3 用例 12

输入账号：15427042 输入密码： 1234512345123451234 51

预期输出：密码长度超过 20 实际输出

表 3 用例 13

输入账号：未测试 输入密码： 123456

预期输出：用户名超过最大字长 实际输出

表 3 用例 14

输入账号：1542704 输入密码： 22222222222222222222

预期输出：账号错误 实际输出

表 3 用例 15

输入账号：154 2 输入密码： 123456

预期输出：用户名含有特殊字符 实际输出

表 3 用例 16

输入账号：15427042 输入密码： 123 6

预期输出：密码错误 实际输出

表 3 用例 17

输入账号：154270 2 输入密码： 1234 6

预期输出：用户名含有特殊字符 实际输出

表 3 用例 18

输入账号：15427.42 输入密码：123456

预期输出：用户名含有特殊字符 实际输出

表 3 用例 19

输入账号：15427042 输入密码：1234.6

预期输出：密码错误 实际输出

表 3 用例 20

输入账号：15470..2 输入密码：123..6

预期输出：用户名含有特殊字符 实际输出

表 3 用例 21

输入账号：管理员 输入密码： 123456

预期输出：用户名不能含有汉字 实际输出 账号错误

表 3 用例 22

输入账号：管理员 输入密码： 12345678

预期输出：用户名不能含有汉字 实际输出

1.3 决策表用例测试结果

表 5 用例 1

输入账号：!@#$%^&* 输入密码： 123456

预期输出：用户名含有特殊字符 实际输出

表 5 用例 2

输入账号：******** 输入密码： 123456

预期输出：用户名含有特殊字符 实际输出

表 5 用例 3

输入账号：15427042 输入密码：123456

预期输出：登录成功 实际输出

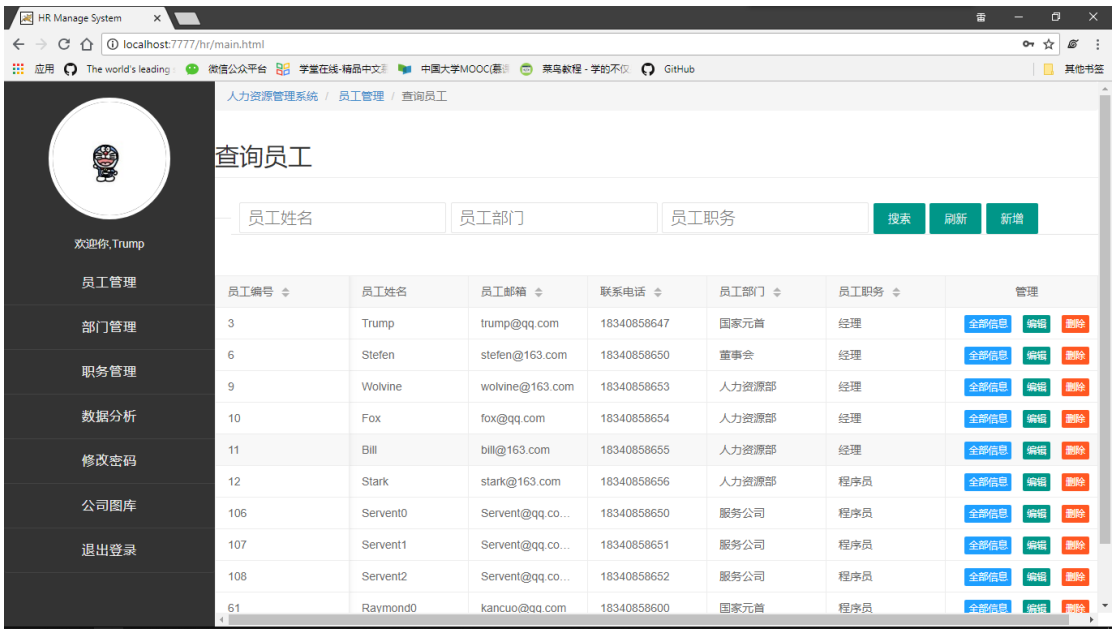


表 5 用例

4 输入账号：1122232432422543 输入密码： 12345

预期输出：用户名错误 实际输出

表 5 用例 5

输入账号：15427042 输入密码： 123 6

预期输出：密码错误 实际输出

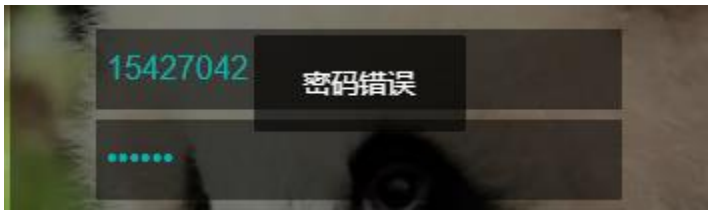


表 5 用例 6

输入账号：15427042 输入密码： 12345

预期输出：密码错误 实际输出

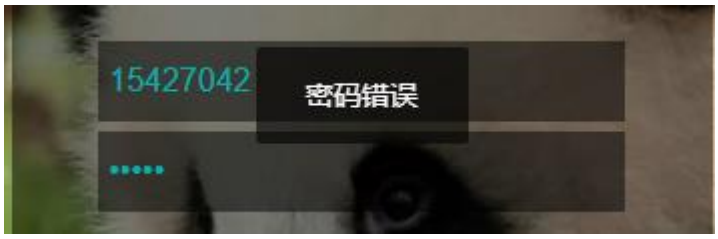


表 5 用例 7

输入账号：!@#$%^&* 输入密码： 11

预期输出：用户名含有特殊字符 实际输出

表 5 用例 8

输入账号：15427099 输入密码： 123456

预期输出：查无此人 实际输出

2 白盒测试结果

2.1 语句覆盖执行结果

表 6 用例 1

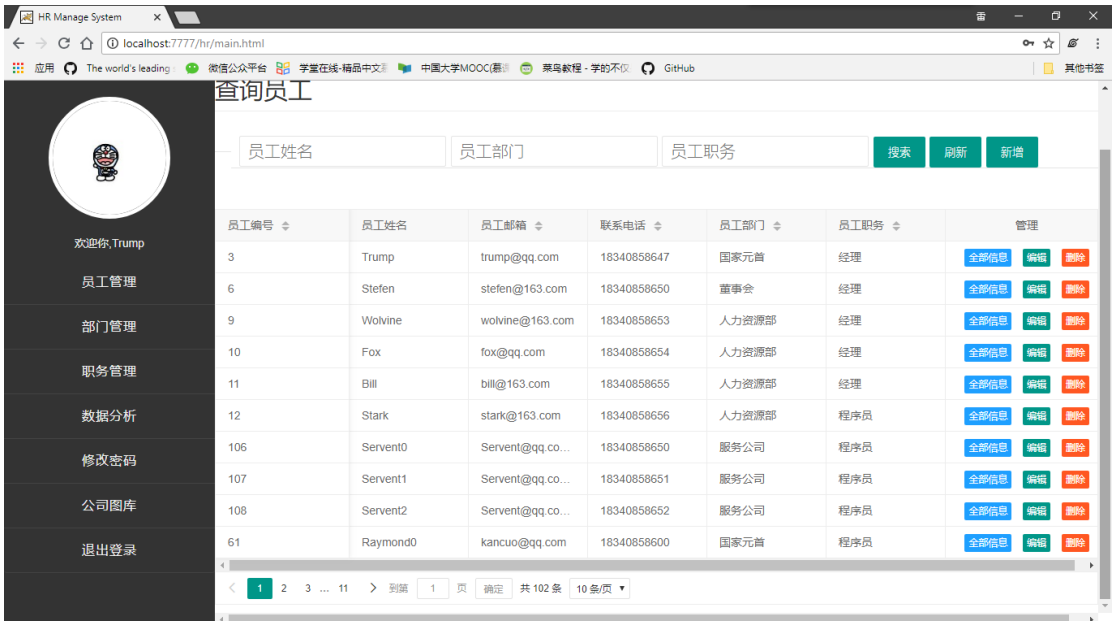
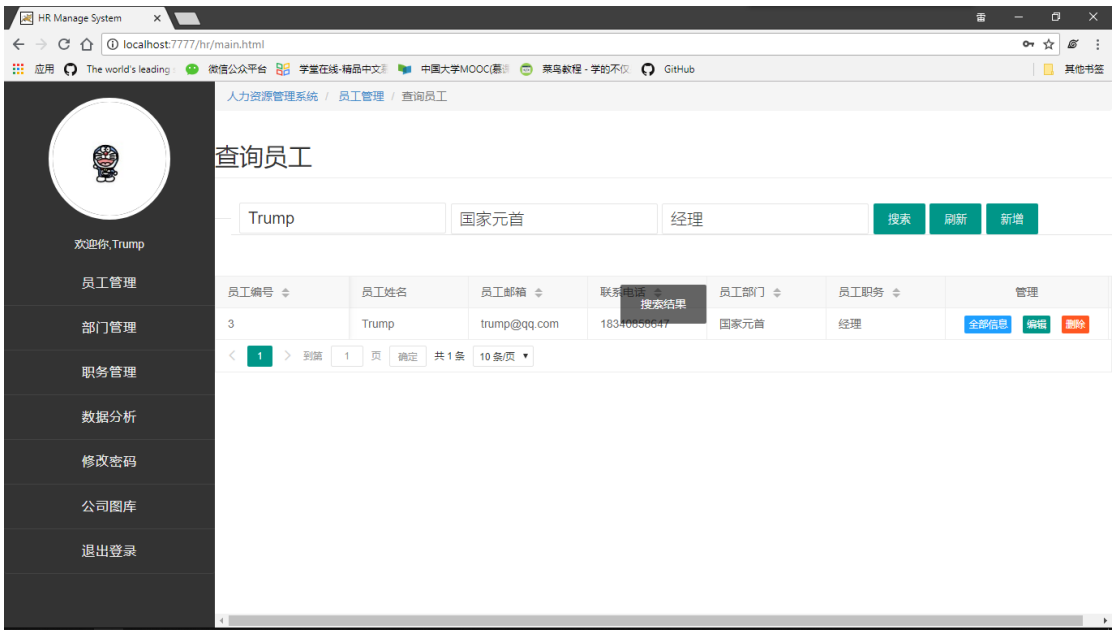


表 6 用例 2



2.2 判定覆盖执行结果

表 7 用例 1

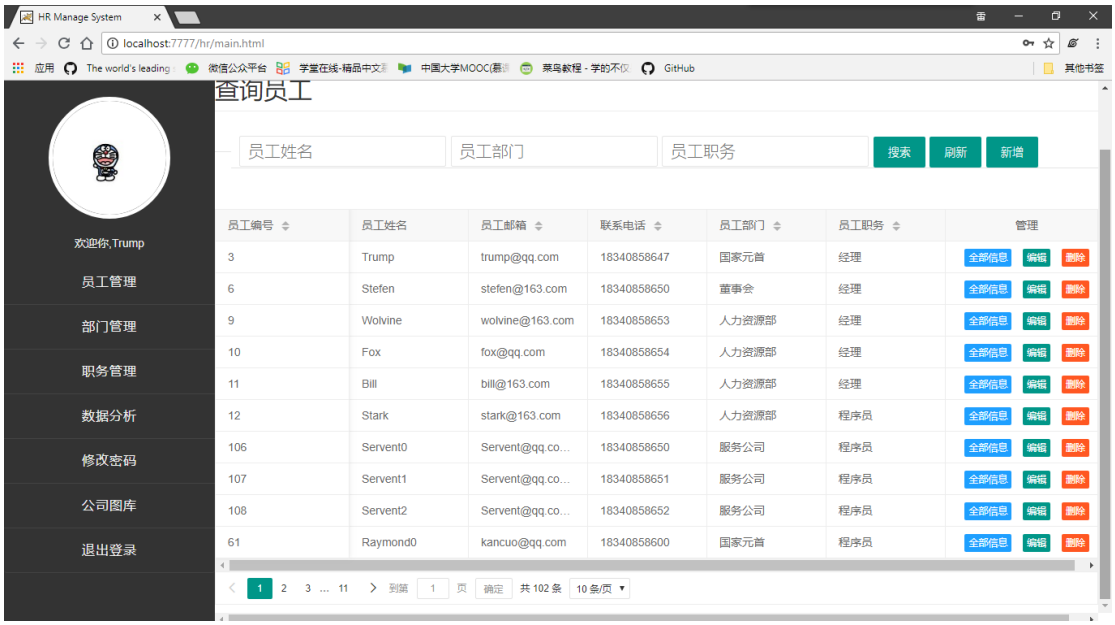


表 7 用例 2

表 7 用例 3

表 7 用例 4

表 7 用例 5

表 7 用例 6

表 7 用例 7

表 7 用例 8

2.3 条件覆盖执行结果

表 8 用例 1

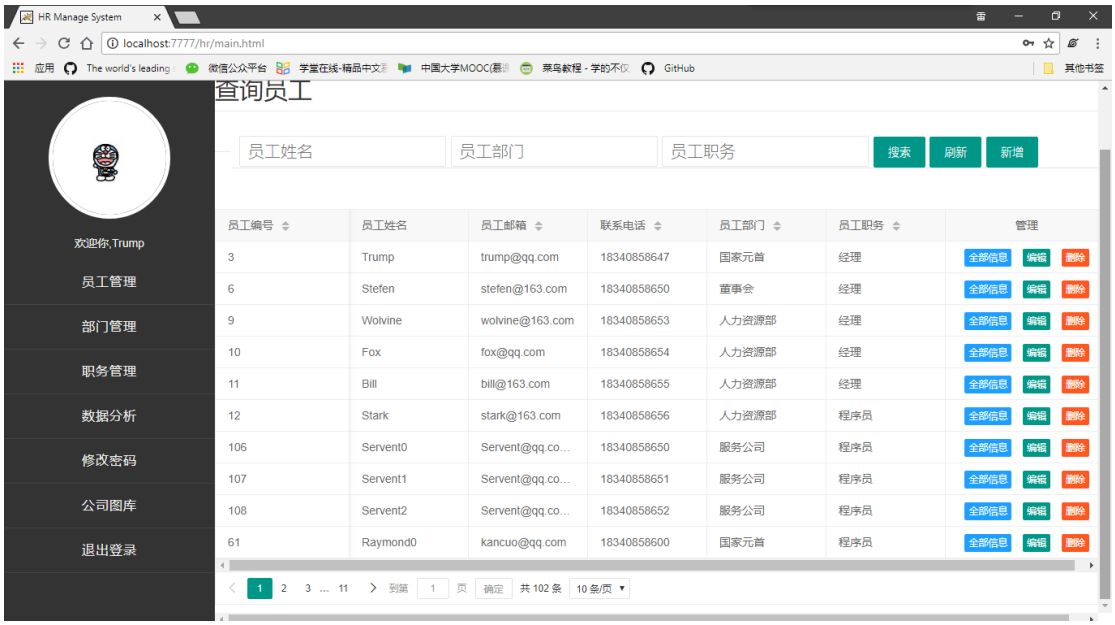


表 8 用例 2

表 8 用例 3

表 8 用例 4

表 8 用例 5

表 8 用例 6

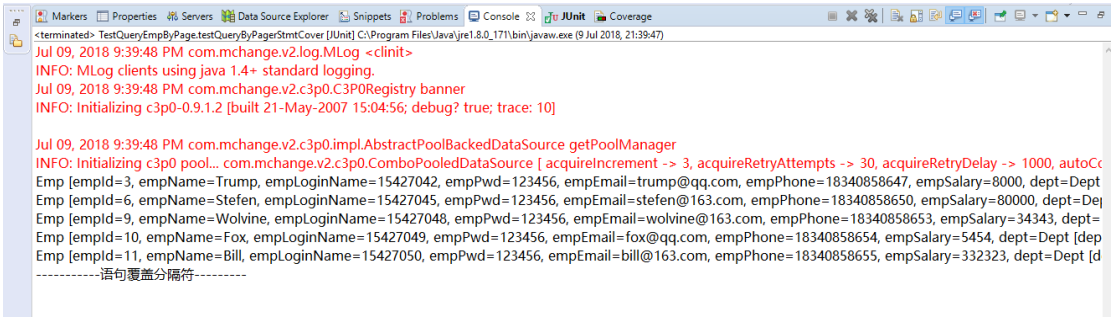
表 8 用例 7

表 8 用例 8

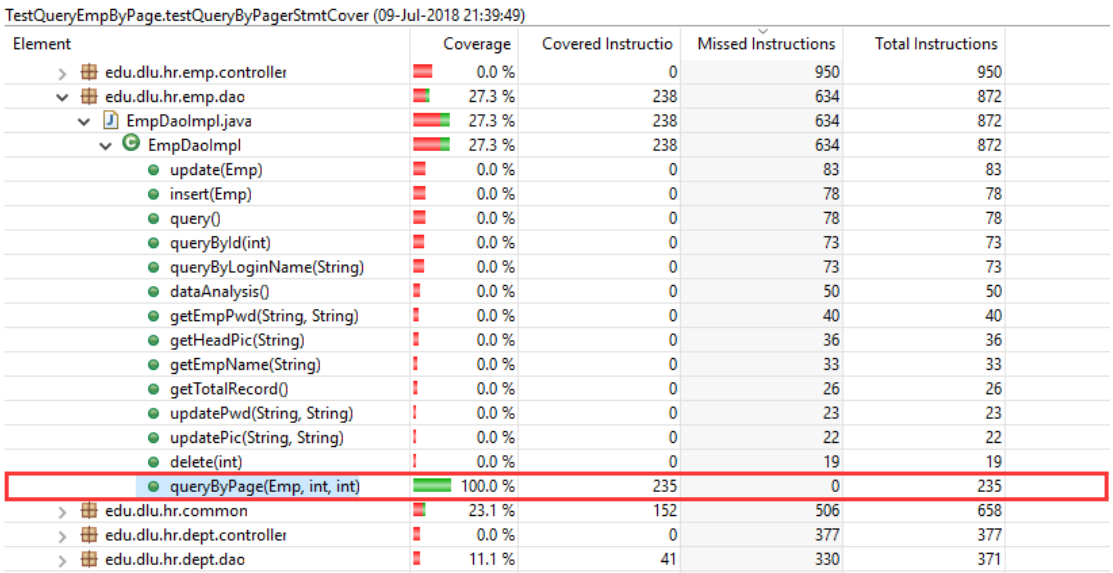
3 单元测试结果

3.1 语句覆盖测试

3.1.1 测试结果



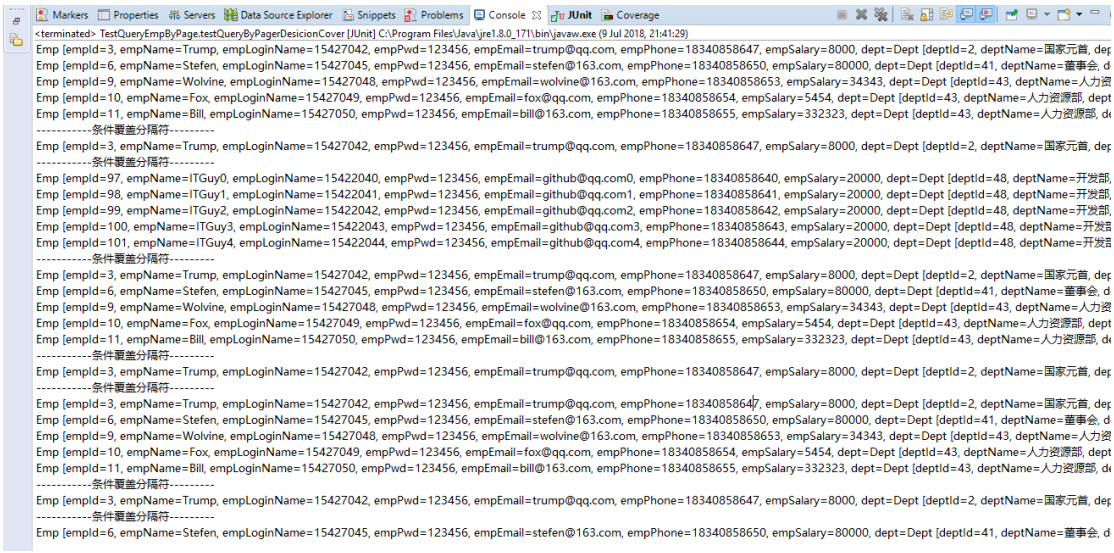
3.1.2 覆盖率



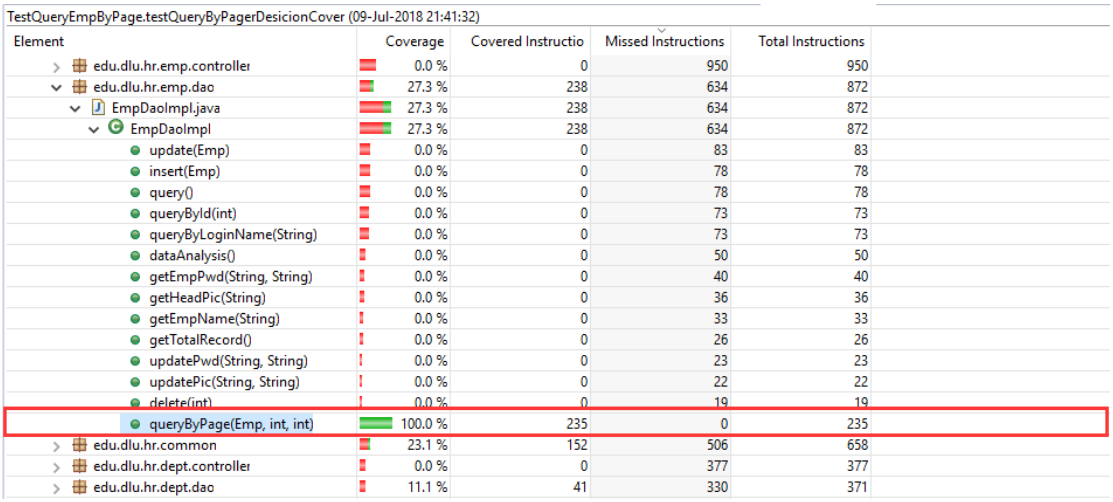
Element	Coverage	Covered Instructio	Missed Instructions	Total Instructions
edu.dlu.hr.emp.controller	0.0 %	0	950	950
edu.dlu.hr.emp.dao	27.3 %	238	634	872
EmpDaoImpl.java	27.3 %	238	634	872
EmpDaoImpl	27.3 %	238	634	872
update(Emp)	0.0 %	0	83	83
insert(Emp)	0.0 %	0	78	78
query()	0.0 %	0	78	78
queryById(int)	0.0 %	0	73	73
queryByLoginName(String)	0.0 %	0	73	73
dataAnalysis()	0.0 %	0	50	50
getEmpPwd(String, String)	0.0 %	0	40	40
getHeadPic(String)	0.0 %	0	36	36
getEmpName(String)	0.0 %	0	33	33
getTotalRecord()	0.0 %	0	26	26
updatePwd(String, String)	0.0 %	0	23	23
updatePic(String, String)	0.0 %	0	22	22
delete(int)	0.0 %	0	19	19
queryByPage(Emp, int, int)	100.0 %	235	0	235
edu.dlu.hr.common	23.1 %	152	506	658
edu.dlu.hr.dept.controller	0.0 %	0	377	377
edu.dlu.hr.dept.dao	11.1 %	41	330	371

3.2 判定覆盖测试

3.2.1 测试结果



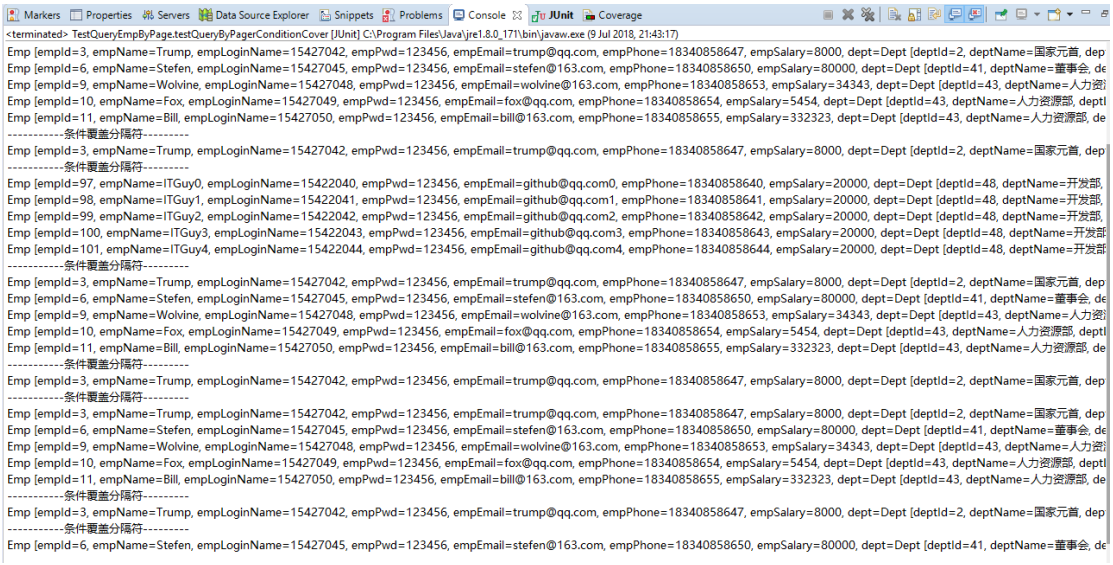
3.2.2 覆盖率



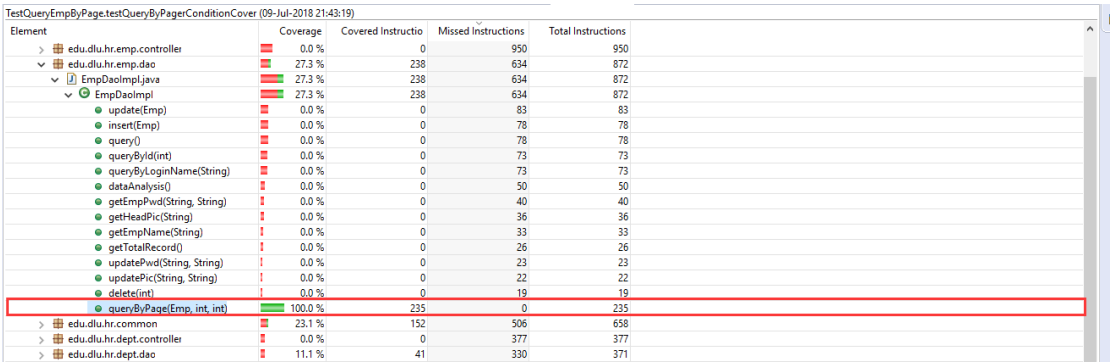
Element	Coverage	Covered Instructio	Missed Instructions	Total Instructions
edu.dlu.hr.emp.controller	0.0 %	0	950	950
edu.dlu.hr.emp.dao	27.3 %	238	634	872
EmpDaoImpl.java	27.3 %	238	634	872
EmpDaoImpl	27.3 %	238	634	872
update(Emp)	0.0 %	0	83	83
insert(Emp)	0.0 %	0	78	78
query()	0.0 %	0	78	78
queryById(int)	0.0 %	0	73	73
queryByLoginName(String)	0.0 %	0	73	73
dataAnalysis()	0.0 %	0	50	50
getEmpPwd(String, String)	0.0 %	0	40	40
getHeadPic(String)	0.0 %	0	36	36
getEmpName(String)	0.0 %	0	33	33
getTotalRecord()	0.0 %	0	26	26
updatePwd(String, String)	0.0 %	0	23	23
updatePic(String, String)	0.0 %	0	22	22
delete(int)	0.0 %	0	19	19
queryByPage(Emp, int, int)	100.0 %	235	0	235
edu.dlu.hr.common	23.1 %	152	506	658
edu.dlu.hr.dept.controller	0.0 %	0	377	377
edu.dlu.hr.dept.dao	11.1 %	41	330	371

3.3 条件覆盖测试

3.3.1 测试结果

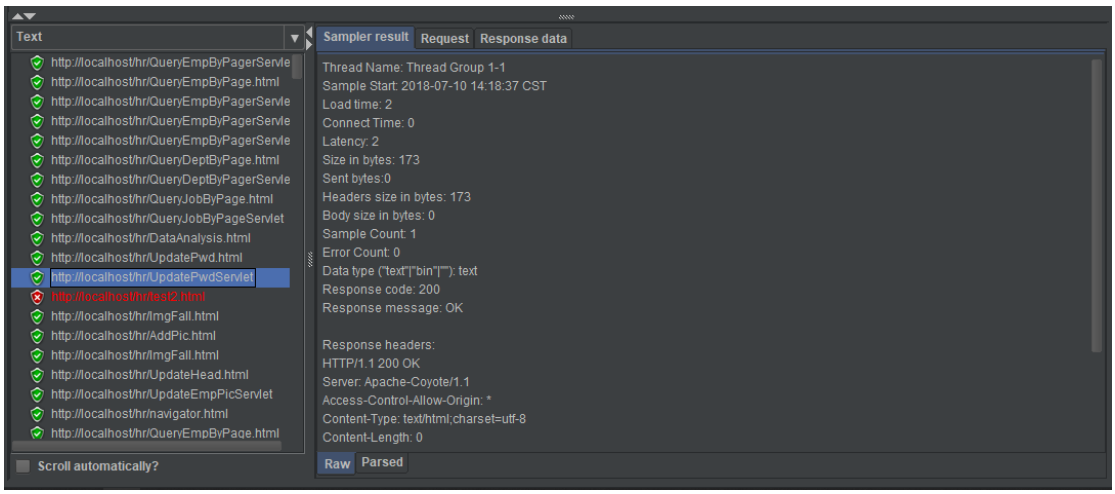


3.3.2 覆盖率

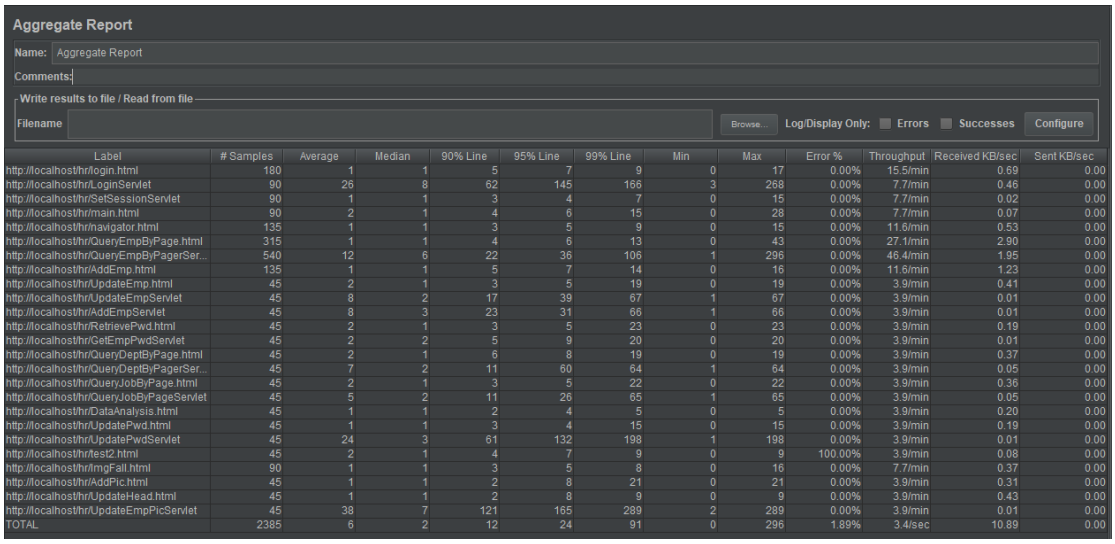


4 自动化测试结果

4.1 查看结果树



4.2 聚集报告



Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	Received KB/sec	Sent KB/sec
http://localhost/hr/login.html	180	1	1	5	7	9	0	17	0.00%	15.5/min	0.69	0.00
http://localhost/hr/LoginServlet	90	26	8	62	145	166	3	268	0.00%	7.7/min	0.46	0.00
http://localhost/hr/SetSessionServlet	90	1	1	3	4	7	0	15	0.00%	7.7/min	0.02	0.00
http://localhost/hr/main.html	90	2	1	4	6	15	0	28	0.00%	7.7/min	0.07	0.00
http://localhost/hr/navigator.html	135	1	1	3	5	9	0	15	0.00%	11.6/min	0.53	0.00
http://localhost/hr/QueryEmpByPage.html	315	1	1	4	6	13	0	43	0.00%	27.1/min	2.90	0.00
http://localhost/hr/QueryEmpByPagerSer...	540	12	6	22	36	106	1	296	0.00%	46.4/min	1.95	0.00
http://localhost/hr/AddEmp.html	135	1	1	5	7	14	0	16	0.00%	11.6/min	1.23	0.00
http://localhost/hr/UpdateEmp.html	45	2	1	3	5	19	0	19	0.00%	3.9/min	0.41	0.00
http://localhost/hr/UpdateEmpServlet	45	8	2	17	39	67	1	67	0.00%	3.9/min	0.01	0.00
http://localhost/hr/AddEmpServlet	45	8	3	23	31	66	1	66	0.00%	3.9/min	0.01	0.00
http://localhost/hr/RetrievePwd.html	45	2	1	3	5	23	0	23	0.00%	3.9/min	0.19	0.00
http://localhost/hr/GetEmpPwdServlet	45	2	2	5	9	20	0	20	0.00%	3.9/min	0.01	0.00
http://localhost/hr/QueryDeptByPage.html	45	2	1	6	8	19	0	19	0.00%	3.9/min	0.37	0.00
http://localhost/hr/QueryDeptByPagerSer...	45	7	2	11	60	64	1	64	0.00%	3.9/min	0.05	0.00
http://localhost/hr/QueryJobByPage.html	45	2	1	3	5	22	0	22	0.00%	3.9/min	0.36	0.00
http://localhost/hr/QueryJobByPageServlet	45	5	2	11	26	65	1	65	0.00%	3.9/min	0.05	0.00
http://localhost/hr/DataAnalysis.html	45	1	1	2	4	5	0	5	0.00%	3.9/min	0.20	0.00
http://localhost/hr/UpdatePwd.html	45	1	1	3	4	15	0	15	0.00%	3.9/min	0.19	0.00
http://localhost/hr/UpdatePwdServlet	45	24	3	61	132	198	1	198	0.00%	3.9/min	0.01	0.00
http://localhost/hr/test2.html	45	2	1	4	7	9	0	9	100.00%	3.9/min	0.08	0.00
http://localhost/hr/ImgFall.html	90	1	1	3	5	8	0	16	0.00%	7.7/min	0.37	0.00
http://localhost/hr/AddPic.html	45	1	1	2	8	21	0	21	0.00%	3.9/min	0.31	0.00
http://localhost/hr/UpdateHead.html	45	1	1	2	8	9	0	9	0.00%	3.9/min	0.43	0.00
http://localhost/hr/UpdateEmpPicServlet	45	38	7	121	165	289	2	289	0.00%	3.9/min	0.01	0.00
TOTAL	2385	6	2	12	24	91	0	296	1.89%	3.4/sec	10.89	0.00

4.3 总结报告

Summary Report

Name: Summary Report

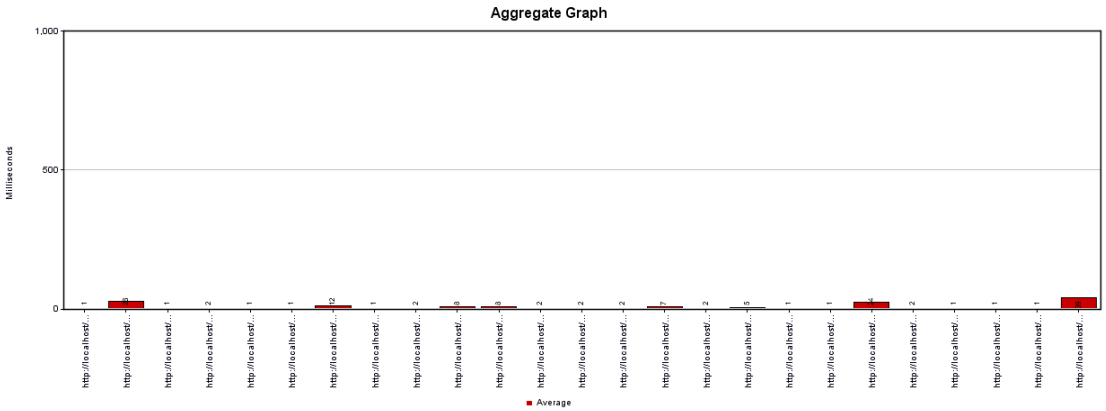
Comments:

Write results to file / Read from file

Filename | Browse... | Log/Display Only: Errors Successes | Configure

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
http://localhost/hr/login.html	180	1	0	17	2.39	0.00%	15.5/min	0.69	0.00	2721.1
http://localhost/hr/LoginServlet	90	26	3	268	45.57	0.00%	7.7/min	0.46	0.00	3629.5
http://localhost/hr/SetSessionServlet	90	1	0	15	1.89	0.00%	7.7/min	0.02	0.00	173.0
http://localhost/hr/main.html	90	2	0	28	3.61	0.00%	7.7/min	0.07	0.00	573.0
http://localhost/hr/navigator.html	135	1	0	15	2.00	0.00%	11.6/min	0.53	0.00	2793.3
http://localhost/hr/QueryEmpByPage.html	315	1	0	43	3.71	0.00%	27.1/min	2.90	0.00	6578.0
http://localhost/hr/QueryEmpByPagerServlet	540	12	1	296	22.90	0.00%	46.4/min	1.95	0.00	2584.4
http://localhost/hr/AddEmp.html	135	1	0	16	2.63	0.00%	11.6/min	1.23	0.00	6489.0
http://localhost/hr/UpdateEmp.html	45	2	0	19	3.08	0.00%	3.9/min	0.41	0.00	6443.4
http://localhost/hr/UpdateEmpServlet	45	8	1	67	13.34	0.00%	3.9/min	0.01	0.00	173.0
http://localhost/hr/AddEmpServlet	45	8	1	66	12.26	0.00%	3.9/min	0.01	0.00	173.4
http://localhost/hr/RetrievePwd.html	45	2	0	23	3.33	0.00%	3.9/min	0.19	0.00	2996.8
http://localhost/hr/GetEmpPwdServlet	45	2	0	20	3.76	0.00%	3.9/min	0.01	0.00	179.0
http://localhost/hr/QueryDeptByPage.html	45	2	0	19	3.28	0.00%	3.9/min	0.37	0.00	5818.0
http://localhost/hr/QueryDeptByPagerServlet	45	7	1	64	15.17	0.00%	3.9/min	0.05	0.00	798.0
http://localhost/hr/QueryJobByPage.html	45	2	0	22	3.50	0.00%	3.9/min	0.36	0.00	5709.0
http://localhost/hr/QueryJobByPageServlet	45	5	1	65	12.17	0.00%	3.9/min	0.05	0.00	780.0
http://localhost/hr/DataAnalysis.html	45	1	0	5	0.99	0.00%	3.9/min	0.20	0.00	3181.4
http://localhost/hr/UpdatePwd.html	45	1	0	15	2.32	0.00%	3.9/min	0.19	0.00	3014.0
http://localhost/hr/UpdatePwdServlet	45	24	1	198	45.31	0.00%	3.9/min	0.01	0.00	173.0
http://localhost/hr/test2.html	45	2	0	9	2.01	100.00%	3.9/min	0.08	0.00	1225.0
http://localhost/hr/ImgFall.html	90	1	0	16	2.20	0.00%	7.7/min	0.37	0.00	2957.2
http://localhost/hr/AddPic.html	45	1	0	21	3.29	0.00%	3.9/min	0.31	0.00	4886.0
http://localhost/hr/UpdateHead.html	45	1	0	9	2.05	0.00%	3.9/min	0.43	0.00	6798.4
http://localhost/hr/UpdateEmpPicServlet	45	38	2	289	62.33	0.00%	3.9/min	0.01	0.00	173.0
TOTAL	2385	6	0	296	19.71	1.89%	3.4/sec	10.89	0.00	3263.7

4.4 聚集图表



4.5 表格中查看结果

Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
37	14:11:57.092	Thread Group 1-1	http://localhost/hr/Query...	9		798	0	9	
38	14:11:57.101	Thread Group 1-1	http://localhost/hr/Query...	3		5709	0	3	
39	14:11:57.104	Thread Group 1-1	http://localhost/hr/Query...	5		780	0	5	
40	14:11:57.109	Thread Group 1-1	http://localhost/hr/DataA...	4		3181	0	4	
41	14:11:57.114	Thread Group 1-1	http://localhost/hr/Updat...	5		3014	0	5	
42	14:11:57.120	Thread Group 1-1	http://localhost/hr/Updat...	6		173	0	6	
43	14:11:57.127	Thread Group 1-1	http://localhost/hr/test2...	4		1225	0	4	
44	14:11:57.132	Thread Group 1-1	http://localhost/hr/ImgFa...	2		2957	0	2	
45	14:11:57.134	Thread Group 1-1	http://localhost/hr/AddPi...	3		4886	0	3	
46	14:11:57.137	Thread Group 1-1	http://localhost/hr/ImgFa...	2		2957	0	2	
47	14:11:57.139	Thread Group 1-1	http://localhost/hr/Updat...	9		6798	0	9	
48	14:11:57.148	Thread Group 1-1	http://localhost/hr/Updat...	15		173	0	15	
49	14:11:57.163	Thread Group 1-1	http://localhost/hr/navig...	1		2793	0	1	
50	14:11:57.165	Thread Group 1-1	http://localhost/hr/Query...	1		6578	0	1	
51	14:11:57.166	Thread Group 1-1	http://localhost/hr/Query...	9		3338	0	9	
52	14:11:57.175	Thread Group 1-1	http://localhost/hr/Query...	9		6578	0	9	
53	14:11:57.184	Thread Group 1-1	http://localhost/hr/Query...	6		3338	0	6	
54	14:13:44.966	Thread Group 1-1	http://localhost/hr/login....	3		2721	0	3	
55	14:13:44.970	Thread Group 1-1	http://localhost/hr/Login...	12		3668	0	12	
56	14:13:44.983	Thread Group 1-1	http://localhost/hr/SetSe...	3		173	0	3	
57	14:13:44.986	Thread Group 1-1	http://localhost/hr/main....	4		573	0	4	
58	14:13:44.990	Thread Group 1-1	http://localhost/hr/navig...	2		2793	0	2	
59	14:13:44.993	Thread Group 1-1	http://localhost/hr/Query...	2		6578	0	2	
60	14:13:44.995	Thread Group 1-1	http://localhost/hr/Query...	8		3338	0	8	
61	14:13:45.004	Thread Group 1-1	http://localhost/hr/Query...	2		6578	0	2	
62	14:13:45.006	Thread Group 1-1	http://localhost/hr/Query...	8		3338	0	8	
63	14:13:45.014	Thread Group 1-1	http://localhost/hr/AddE...	2		6489	0	2	
64	14:13:45.017	Thread Group 1-1	http://localhost/hr/Query...	1		6578	0	1	

4.6 图表结果

5 SQL 注入测试结果

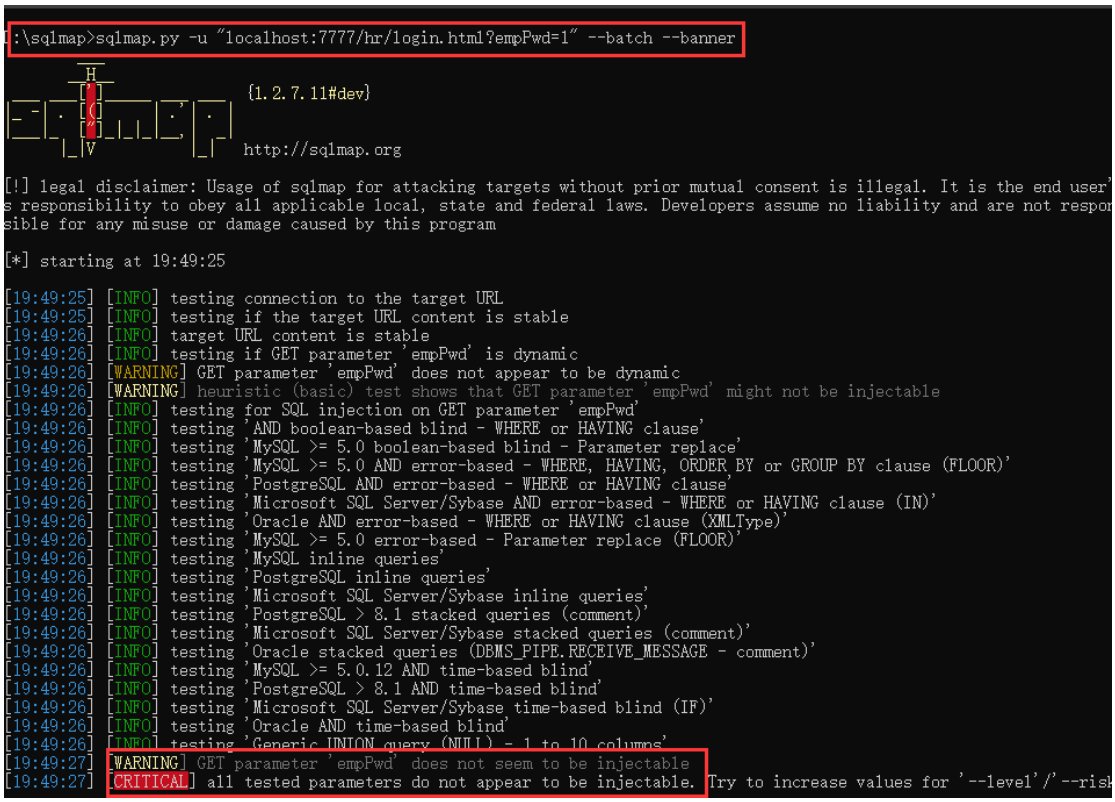
5.1 登录窗口用户名 SQL 注入检测

检测结果：不可注入

```
I:\sqlmap>sqlmap.py -u "localhost:7777/hr/login.html?empLoginName=1" --batch --banner
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.2.7.11#dev}
|_ -| . ["]     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V          |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting at 19:47:09

[19:47:09] [INFO] testing connection to the target URL
[19:47:09] [INFO] testing if the target URL content is stable
[19:47:10] [INFO] target URL content is stable
[19:47:10] [INFO] testing if GET parameter 'empLoginName' is dynamic
[19:47:10] [WARNING] GET parameter 'empLoginName' does not appear to be dynamic
[19:47:10] [WARNING] heuristic (basic) test shows that GET parameter 'empLoginName' might not be injectable
[19:47:10] [INFO] testing for SQL injection on GET parameter 'empLoginName'
[19:47:10] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[19:47:10] [INFO] testing 'MySQL >= 5.0 boolean-based blind - Parameter replace'
[19:47:10] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[19:47:10] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[19:47:10] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[19:47:10] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[19:47:10] [INFO] testing 'MySQL >= 5.0 error-based - Parameter replace (FLOOR)'
[19:47:10] [INFO] testing 'MySQL inline queries'
[19:47:10] [INFO] testing 'PostgreSQL inline queries'
[19:47:10] [INFO] testing 'Microsoft SQL Server/Sybase inline queries'
[19:47:10] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[19:47:10] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[19:47:10] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[19:47:10] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind'
[19:47:10] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[19:47:10] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[19:47:10] [INFO] testing 'Oracle AND time-based blind'
[19:47:10] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[19:47:11] [WARNING] GET parameter 'empLoginName' does not seem to be injectable
[19:47:11] [CRITICAL] all tested parameters do not appear to be injectable. Try to increase values for '--level'/'--risk
```

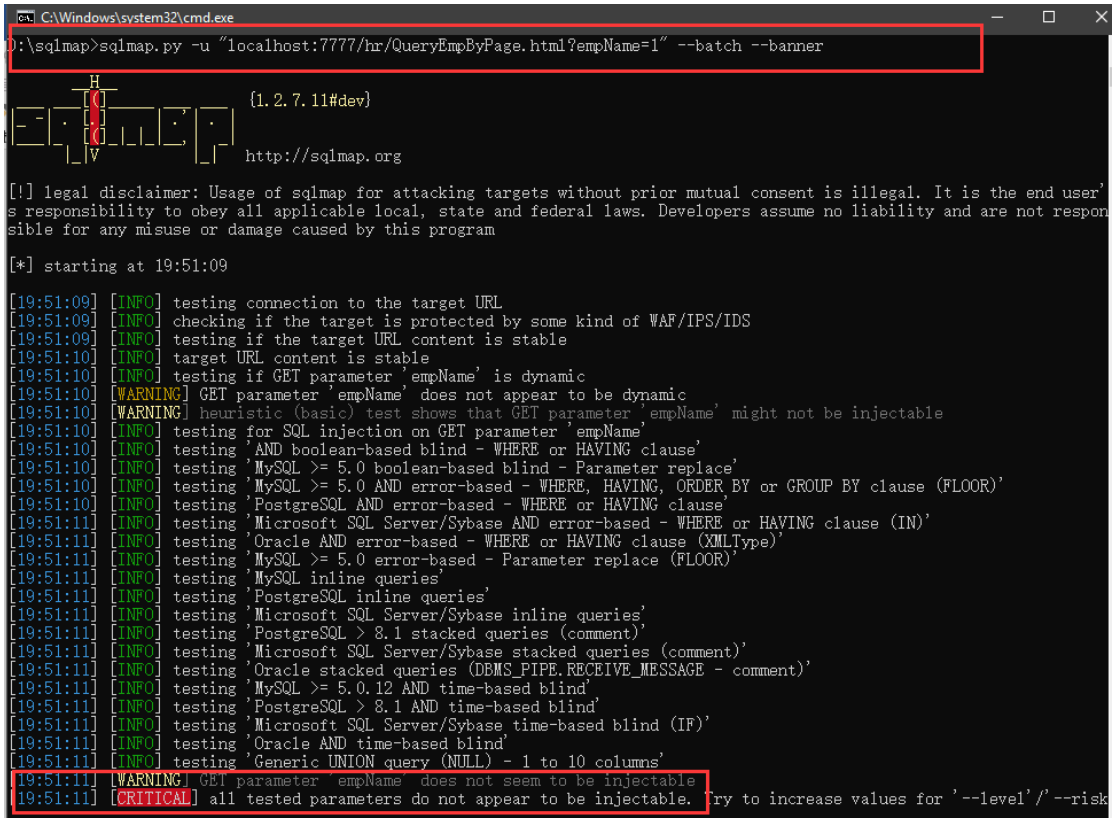
5.2 登录窗口密码 SQL 注入检测

检测结果：不可注入



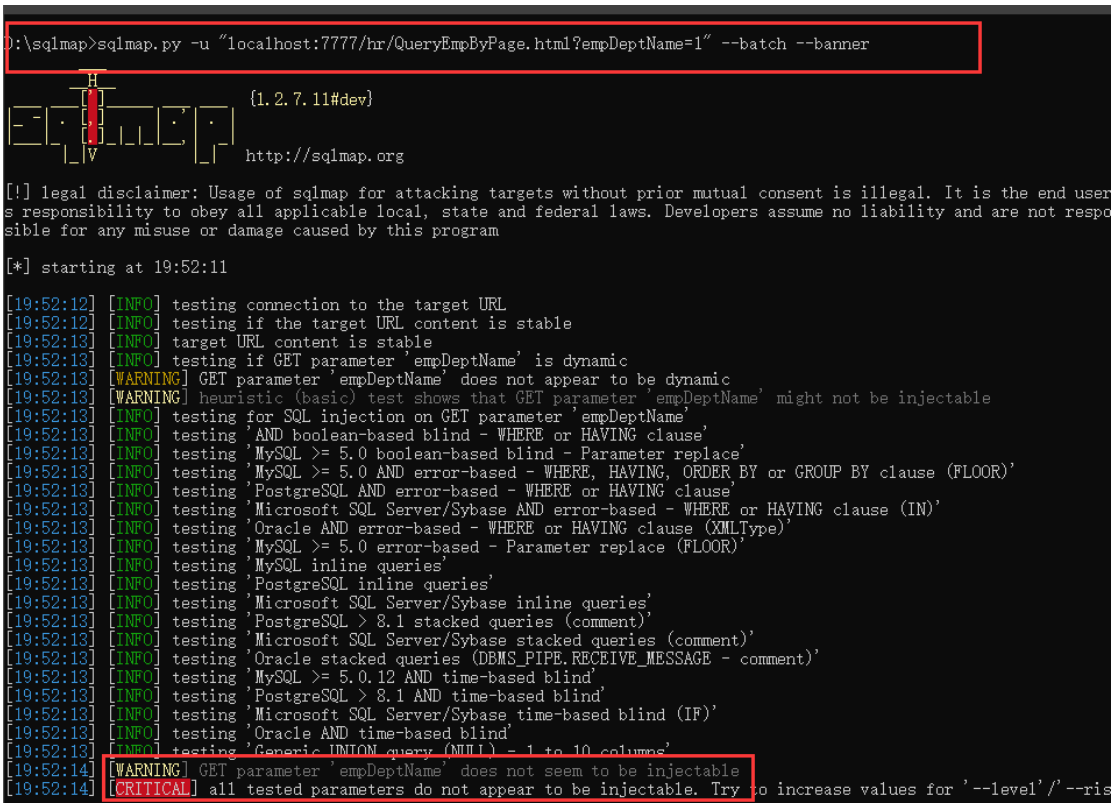
5.3 分页查询页面员工姓名 SQL 注入检测

检测结果：不可注入



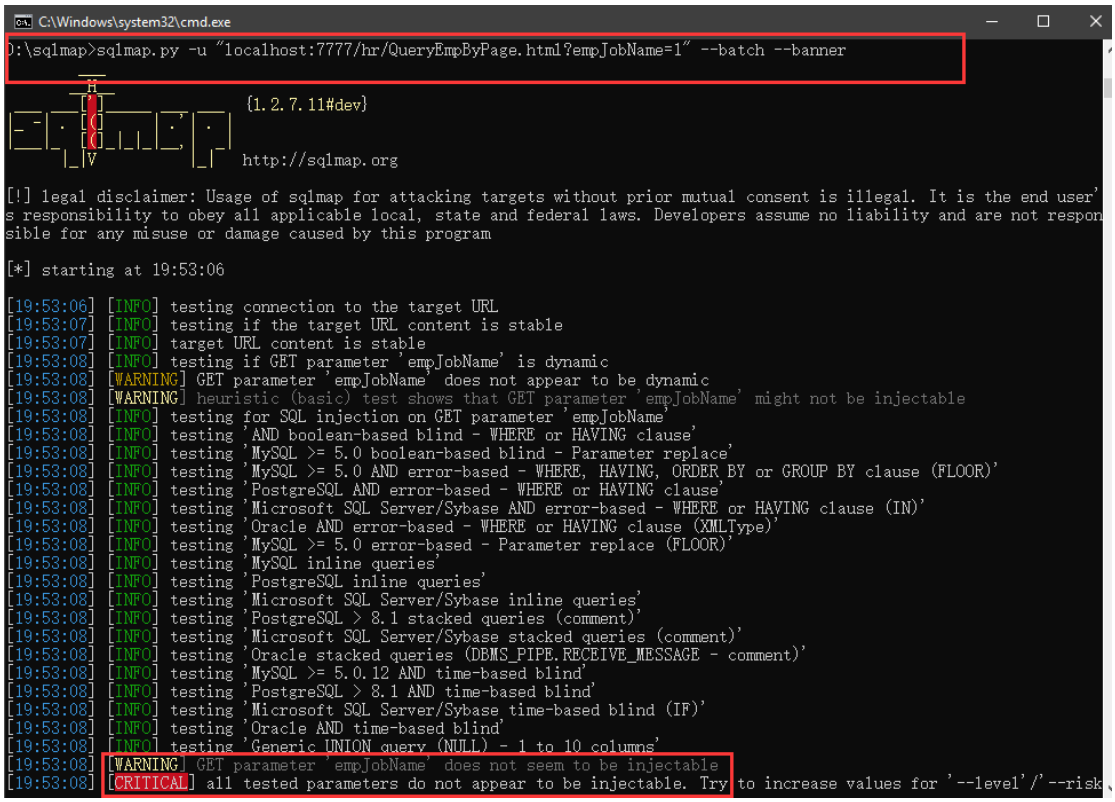
5.4 分页查询页面员工部门 SQL 注入检测

检测结果：不可注入



5.5 分页查询页面员工职务 SQL 注入检测

检测结果：不可注入



五 缺陷报告

1 缺陷概述

软件缺陷简单的来说就是存在于软件之中的那些不希望，或不可接受的偏差，而导致软件产生质量问题。本失物招领系统主要是拾物者和失主之间的一种交互，建立的一个平台，软件的缺陷主要包含首页的部分超级链接无效等。

2 BUG 统计

缺陷编号	缺陷模块	缺陷描述
1	登录窗口	1 注册时限制用户名长度为 8 位，但是登录时却没有限制登录用户名的长度； 2 错误提示信息笼统单一 3 错误登录次数未加限制，可被暴力破解攻击 4 未登录状态直接使用 URL 进入主界面可以进行所有操作 5 在输入用户名与密码时每次键盘输入一个字符就会调用 KeyDown()函数，而此函数只应在登录信息全部录入点击登录时才应调用。
2	找回密码	在用户名均正确时可以找回密码，但是当用户名或密码存在错误时没有任何提示信息，也没有任何返回结果

3	增加员工	1 电子邮箱只要含有@便可注册，未加具体验证 2 员工姓名未加验证 3 初始密码与确认密码为空仍可注册，未加具体验证 4 电话号码只要满足 11 位便可注册，未加具体验证 5 注册信息未加单一性验证,数据库存在仍可添加成功
4	图片管理	1 初次登录时未显示默认头像，导致头像去不能点击，不能更改头像 2 当在测试类中删除图片时,图库中仍有图片信息残余且无法删除 3 公司图库只可看不可进行操作
5	员工管理	1 未分权限,导致管理员登录时显示自身信息且可删除自身 2 点击员工记录的“编辑”按钮时弹出界面存在脏数据以及无效链接 3 点击员工记录的“编辑”按钮时弹出界面存在跳转链接混乱 4 员工记录某些属性可以直接更改,但无法对数据进行修改 5 点击新增按钮，弹出“增加员工界面”，单击“回到登录界面”按钮跳转到主页，提示误导，链接出错 6 分页显示出错，当前有 128 条数据，页面大小为 10，第 13 页应显 8 条数据,，实显 10 条数据 7 主页面顶部链接失效 8 进行精确查询后点击删除未停留当前界面,跳回到主界面
6	部门管理	1 点击部门记录的“编辑”按钮时弹出界面存在脏数据以及无效链接 2 点击部门记录的“编辑”按钮时弹出界面存在跳转链接混乱 3 分页显示出错，当前有 13 条数据，页面大小为 10，第 2 页应显 3 条数据，实显 4 条数据 4 删除部门时显示删除成功但是并未修改数据信息,未提示当前部门有员工存在 5 进行精确查询后点击删除未停留当前界面，跳回到主界面
7	职务管理	1 点击职务记录的“编辑”按钮时弹出界面存在脏数据以及无效链接 2 点击职务记录的“编辑”按钮时弹出界面存在跳转链接混乱 3 据员工管理与部门管理推断职务管理分页仍然存在错误 4 删除职务时显示删除成功但是并未修改数据信息,未提示当前职务有员工存在 5 进行精确查询后点击删除未停留当前界面,跳回到主

		界面
8	修改密码	存在失效索引，修改密码成功后跳到脏页面test2.html,此页面实际并不存在。

六 总结与建议

在此次不完全测试中在 8 个模块中发现了攻击 32 个错误，系统处于高危级别，需要大量修改。

在黑盒测试与白盒测试中出现了很多相同的数据，数据略显冗余，但是基本上满足了各种方法的条件，理论上覆盖了所用方法的可能测试用例，但是正如软件测试是检错一样，软件测试本身也会出现错误和纰漏，所以黑盒测试和白盒测试中可能还有需要更加完善的地方。

在 JUnit 单元测试中只测试了分页查询的部分的代码，所写测试类分别为语句覆盖，判定覆盖，条件覆盖，在使用 JUnit 统计覆盖率时由于只调用了一个函数，所以只有该方法名覆盖率为 100%，纵观该函数整体所占的覆盖率则偏低。

在使用自动化测试工具 JMeter 进行自动化测试中对网站的性能与压力测试只是做了简单的测试，因为网站并没有上传到服务器而只是在本地的 Tomcat 中进行测试，所以只是象征性的简单测试。

在使用 SQLMAP 进行 SQL 注入检测时选取用例太少，不具代表性，所选用例不可进行 SQL 注入，并不代表系统没有不可以进行 SQL 注入的地方，所以仍建议进行 SQL 语句的检查以及增加数据库的安全性,防止被 SQL 注入然后被“拖库”，泄露用户信息。

具体的 SQL 注入方法措施包括进行数据有效性检验，封装数据信息，去除代码中的敏感信息，替换或删除单引号，制定错误返回页面，限制 SQL 字符串连接的配置文件，用户权限分离等方式。

在测试过程中并未采取 XSS(Cross-site scripting)检测，这类攻击通常包含在 Html 以及 JavaScript 中,是一种网站应用程序的安全漏洞攻击，属于代码注入的一种。预防的主要措施就是对 HTML 进行过滤。