1 系统测试

本测试的测试目的是教师教学教研绩效考核系统等后台主要操作的基本功能的单元测试。为此，采用 NUnit 测试工具进行单元测试，验证是否满足系统的需求。

这个测试机器的配置环境如下：

操作系统：Micrsoft windowXP Professional SP2

测试工具：NUnit-Net-2.0 2.2.8

CPU：P4 2.8G

内存：512M

硬盘：80G

在 NUnit 工具测试中，绿色表示测试通过；红色表示测试失败；黄色表示测试警告。

数据库基类主要负责整个系统与数据库管理系统的连接、断开、更新、插入、删除数据。

主要测试代码为：

```
[TestFixture]
    public class SqlBaseTest
    {
        public sqlBase operate = null;
        [SetUp]
        public void SetUp()
        {
            operate = sqlBase.getSqlObj();
            operate.Open();
        }
        [TearDown]
        public void Finshed()
        {
            operate.Close();
        }
        [Test]
        public void ExecuteNonQueryIn()
        {
            string name = "大富豪";
            string text = "解放后";
            string sql = "insert into test (name,sex) values('" + name + "','" + text + "')";
            int i = operate.ExecuteNonQuerySql(sql);
            Console.WriteLine(i);
        }.....
```

对数据库基类的单元测试测试：

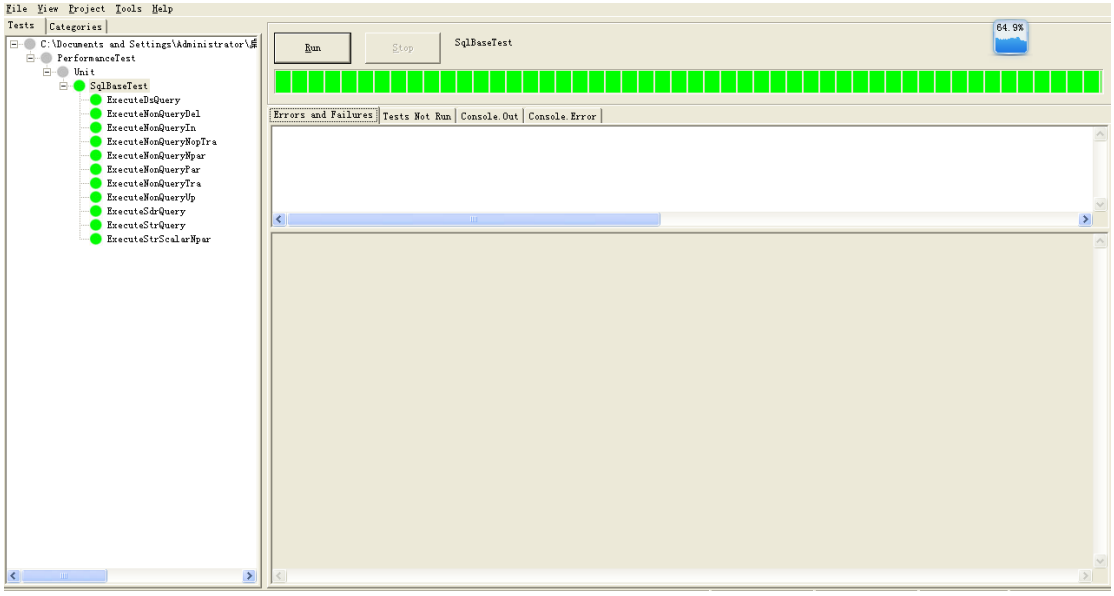

图 4-1 数据库基类测试图

查看教师所填的考核信息时，常常需要获取某个数据表的实例化对象实体。

主要测试代码：

```
[TestFixture]
public class get_TTR
{
    public sqlBase sqlControl = null;
    [SetUp]
    public void SetUp()
    {
        sqlControl = sqlBase.getSqlObj();
        sqlControl.Open();
    }
    [TearDown]
    public void Finshed()
    {
        sqlControl.Close();
    }
    [Test]
    public void ExecuteNonQueryIn()
    {
        string sql = "select * from TTR_grhj where grhjid=1";
        sqlControl.Open();
        DataTable dt = sqlControl.ExecuteDataTable(sql);
        sqlControl.Close();
        TTR_grhj grhj = new TTR_grhj();
        grhj.grhjid = Convert.ToInt32(dt.Rows[0]["grhjid"].ToString());
```

```
            grhj.assessid = dt.Rows[0]["assessid"].ToString();
            grhj.grhjtid = dt.Rows[0]["grhjtid"].ToString();
            grhj.gjwnum = Convert.ToInt32(dt.Rows[0]["gjwnum"].ToString());
            grhj.gjwpeonum = Convert.ToInt32(dt.Rows[0]["gjwpeonum"].ToString());
            grhj.gjwtxt = dt.Rows[0]["gjwtxt"].ToString();
            grhj.sjwnum = Convert.ToInt32(dt.Rows[0]["sjwnum"].ToString());
            grhj.sjwpeonum = Convert.ToInt32(dt.Rows[0]["sjwpeonum"].ToString());
            grhj.sjwtxt = dt.Rows[0]["sjwtxt"].ToString();
            grhj.ssyjnum = Convert.ToInt32(dt.Rows[0]["ssyjnum"].ToString());
            grhj.ssyjpeonum = Convert.ToInt32(dt.Rows[0]["ssyjpeonum"].ToString());
            grhj.ssyjtxt = dt.Rows[0]["ssyjtxt"].ToString();
            grhj.xyxjnum = Convert.ToInt32(dt.Rows[0]["xyxjnum"].ToString());
            grhj.xyxjpeonum = Convert.ToInt32(dt.Rows[0]["xyxjpeonum"].ToString());
            grhj.xyxjtxt = dt.Rows[0]["xyxjtxt"].ToString();
            grhj.xjsjnum = Convert.ToInt32(dt.Rows[0]["xjsjnum"].ToString());
            grhj.xjsjpeonum = Convert.ToInt32(dt.Rows[0]["xjsjpeonum"].ToString());
            grhj.xjsjtxt = dt.Rows[0]["xjsjtxt"].ToString();
            grhj.jxgknum = Convert.ToInt32(dt.Rows[0]["jxgknum"].ToString());
            grhj.jxgkpeonum = Convert.ToInt32(dt.Rows[0]["jxgkpeonum"].ToString());
            grhj.jxgktxt = dt.Rows[0]["jxgktxt"].ToString();
            grhj.image = 0;
        }

        }
```

对获取实体类的单元测试：

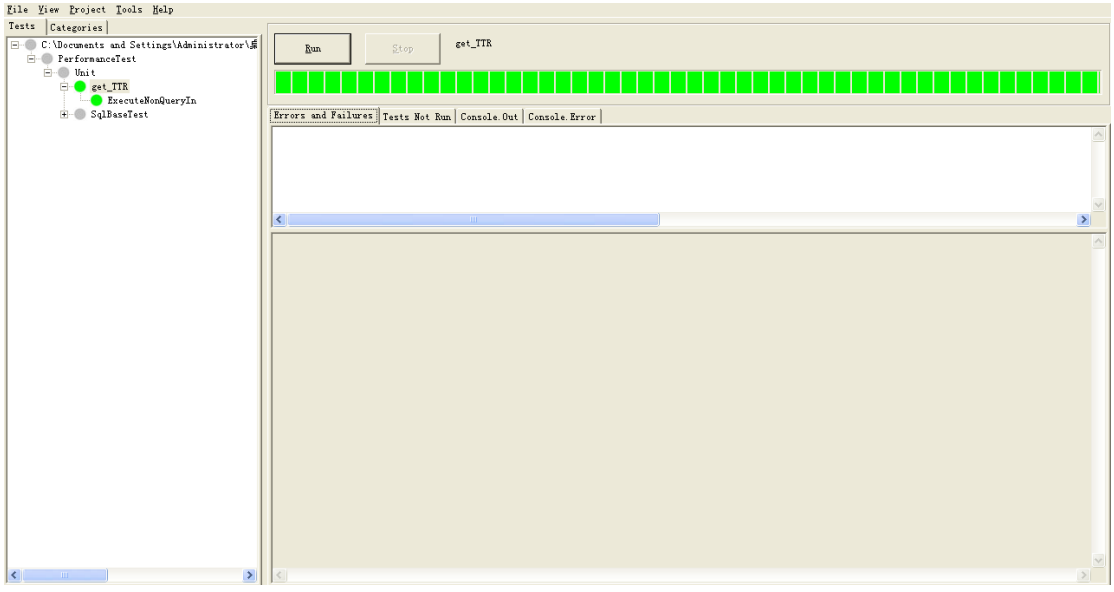

图 4-2 获取实体类测试图

在教师考核成绩的审核中常常需要进行审核通过操作。

主要测试代码：

```
[TestFixture]
    public class Surego
```

```
{
    public sqlBase sqlControl = null;

    [SetUp]
    public void SetUp()
    {
        sqlControl = sqlBase.getSqlObj();
        sqlControl.Open();
    }

    [TearDown]
    public void Finshed()
    {
        sqlControl.Close();
    }

    [Test]
    public void ExecuteNonQueryIn()
    {
        SqlParameter[] sqlparam = new SqlParameter[1];
        sqlparam[0] = new SqlParameter("@grhjid", SqlDbType.Int);
        sqlparam[0].Value = 1;
        sqlControl.Open();
        int h = sqlControl.ExecuteNonQueryP("SuregogrhjExamineList", sqlparam,
CommandType.StoredProcedure);
        sqlControl.Close();
    }
}
```

对教师考核成绩审核类的单元测试：

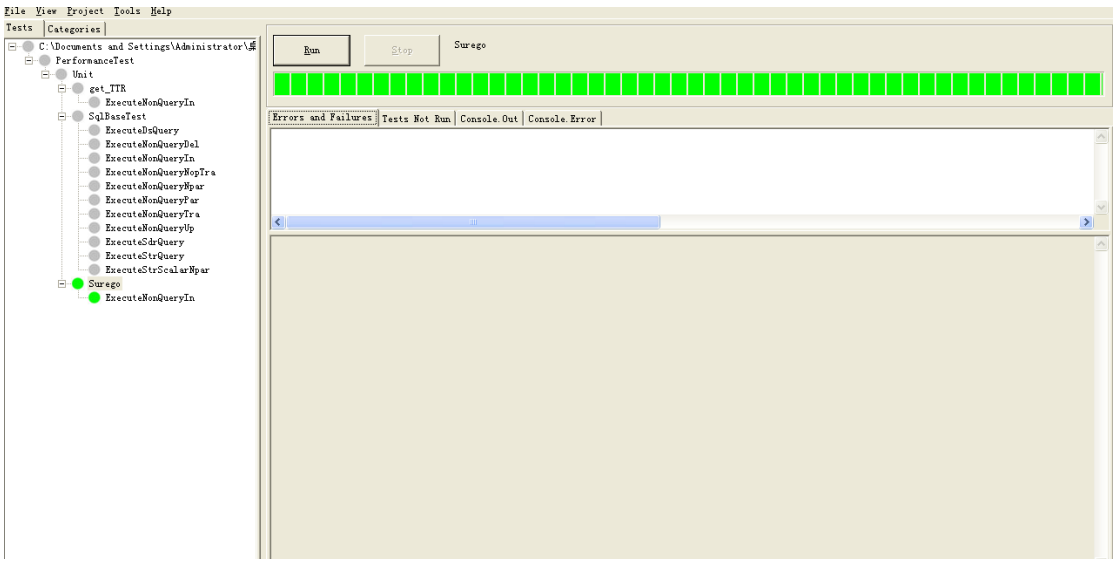

图 4-3 教师考核成绩审核类测试图

在获取数据列表时常常需要进行分页操作。

主要测试代码：

```
[TestFixture]
    public class paging
    {
        public sqlBase sqlControl = null;
        [SetUp]
        public void SetUp()
        {
            sqlControl = sqlBase.getSqlObj();
            sqlControl.Open();
        }
        [TearDown]
        public void Finshed()
        {
            sqlControl.Close();
        }
        [Test]
        public void ExecuteNonQueryIn()
        {
            Int32 RecordCount = 1, RowCount = 1;
            SqlParameter[] sqlparam = new SqlParameter[5];
            sqlparam[0] = new SqlParameter("@SelectOrder", SqlDbType.VarChar, 200);
            sqlparam[0].Value = "order by grhjid asc";
            sqlparam[1] = new SqlParameter("@intPageNo", SqlDbType.Int);
            sqlparam[1].Value = 1;
            sqlparam[2] = new SqlParameter("@intPageSize", SqlDbType.Int);
            sqlparam[2].Value = 2;
            sqlparam[3] = new SqlParameter("@RecordCount", SqlDbType.Int);
            sqlparam[3].Direction = ParameterDirection.Output;
            sqlparam[4] = new SqlParameter("RowCount", SqlDbType.Int);
            sqlparam[4].Direction = ParameterDirection.ReturnValue;

            sqlControl.Open();
            DataSet ds = sqlControl.ExecuteDataSetP("getgrhjExamineList", sqlparam,
CommandType.StoredProcedure);

            RecordCount = (Int32)sqlparam[3].Value; //求出总记录数，该值是output出来的值
            RowCount = (Int32)sqlparam[4].Value;          //求出当前页中的记录数，在最后一页不
等于pagesize，
            sqlControl.Close();

        }
    }
```

对分页类的单元测试：

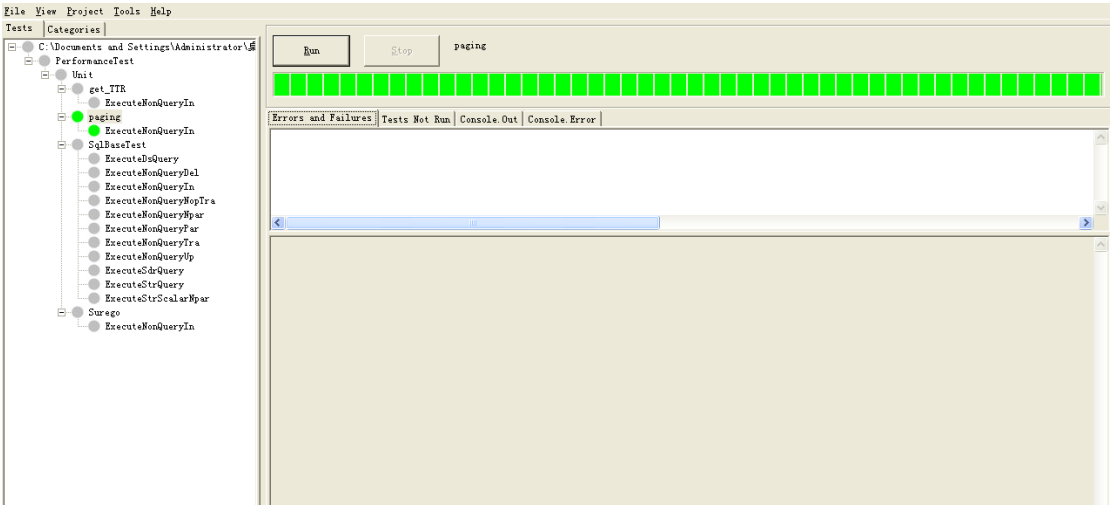

图 4-4 分页类测试图

还可以作些其他测试，关键测试后的性能分析？？

2 系统测试

2.1 测试概述

2.1.1 测试目的

软件测试的目标是以最少的时间和人力，系统地找出软件中潜在的各种错误和缺陷，因此软件测试的目的是：

1. 测试是为了发现程序中的错误而执行程序的过程。
2. 一个好的测试用例在于能够发现至今尚未发现的错误。
3. 一次成功的测试时发现了至今尚未发现的错误测试。

2.1.2 测试原则

在软件测试中，应遵循以下测试原则：

1. 在设计测试用例时，要给出测试的预期结果，便于对照。
2. 在设计测试用例时，不仅要设计合理的输入条件，还要设计不合理的输入条件。
3. 除了检查程序是否做了应该做的工作，还要检查程序是否做了不应该做的工作。
4. 应制定测试计划并严格执行，排除随意性。
5. 长期保持测试用例。

6. 充分注意测试中的群集现象。

2.1.3 测试方法

程序测试有静态测试方法和动态测试方法两类。一般意义上的测试是指动态测试，主要有两种方法，一种是测试产品的功能，称为黑盒测试法；另一种是测试程序内部结构及处理过程称为白盒测试法。

2.2 测试用例

（1）例如：输入数据以下

用户名	密码	登录类型	信息正误
2007051011	123456	研究生	错误
2006051095	123456	管理员	正确

结果错误信息不能登录，正确信息进入管理员界面，如图：

（2）例如：输入以下数据

学号	姓名	事件	奖惩时间	记录时间	备注
2006051095	李四	获校级一等奖学金	2008.12.02	2008.12.05	符合评定标准

经查询结果如下：

成都信息工程学院研究生奖惩信息

事件	奖惩时间	记录时间	备注
校级二等奖学金	2007.12.01	2007.12.03	符合评定标准
校级一等奖学金	2008.12.02	2008.12.05	符合评定标准

制表单位：**成都信息工程学院教务处**　　制表日期：**2010-5-11 16:11:30**

3 测试实验与性能分析

3.1 测试用例

表 4-1 用例分析表

序号	测试模块	测试数据			测试结果
		A	B	C	
1	登陆	已注册的用户名和密码	没有注册的用户名和密码	不做任何输入	登陆成功
2	录入工资	按规定录入工资	非法录入工资	不做任何输入	录入成功
3	录入员工信息	输入正确的员工信息	输入错误的员工信息	不输入	录入成功
4	录入考勤	输入考勤信息	输入错误的考勤信息	不输入	录入成功
5	查看总体工资	点击查看总体工资			查看总体工资
6	查看向银行缴纳金额	点击查看向银行缴纳金额			查看向银行缴纳金额
7	审核科研项目绩效	点击通过审核	不选，通过按钮	输入项目信息删除已通过的审核	操作成功
8	审核学术论文绩效	点击通过审核	不选，通过按钮	删除已通过的审核	操作成功
9	审核获奖成果绩效	点击通过审核	不选，通过按钮	删除已经通过的审核	操作成功
10	审核专利绩效	点击通过审核	不选，通过按钮	删除已经通过的审核	操作成功
11	审核著作绩	点击通过审核	不选，通过按钮	删除已经通	操作成

	效			过的审核	功
12	填写学科建设绩效	输入学科建设绩效信息	输入错误的绩效信息	不输入	录入成功
13	查询工资	输入要查询员工的信息	输入错误的查询信息	不输入	查询成功
14	查询考勤	输入要查询的条件	输入错误的查询信息	不输入	查询成功
15	申请科研项目绩效申请	输入正确的申请信息	输入错误的申请信息	不输入	操作成功
16	申请著作绩效申请	输入正确的申请信息	输入错误的申请信息	不输入	操作成功
17	申请学术论文绩效申请	输入正确的申请信息	输入错误的申请信息	不输入	操作成功
18	申请专利绩效申请	输入正确的申请信息	输入错误的申请信息	不输入	操作成功
19	申请获奖成果绩效申请	输入正确的申请信息	输入错误的申请信息	不输入	操作成功
20	查看申请的绩效奖励是否通过	输入要查询员工的信息	输入错误的查询信息	不输入	操作成功
21	发送邮件	输入正确的邮件信息	输入错误的邮件信息	不输入	发送成功
22	查看邮件	点击查看邮件信息			操作成功

整个系统的功能较多，以上是对主要功能进行的测试用例分析。

3.2 测试效果图

(1)高校工资管理系统的首页

在这个页面里，对网站的操作进行了简要说明，给用户一个大体宏观的操作流程，通过选择“用户”和“管理员”选项来进入网站，进行适合自己身份的操作。图片是以背景的形式插入。首页如图 4-1 所示。

图 4-1 首页图

(2)用户页面。如页面图 4-2。

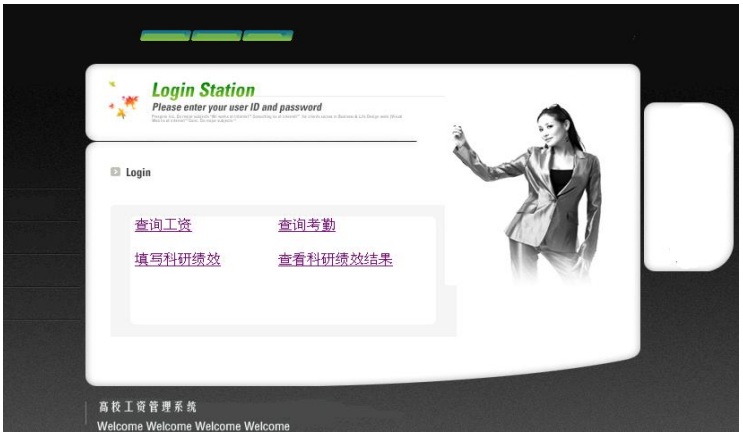

图 4-2 用户页面图

(3)用户查看考勤页面。如页面图 4-3。

图 4-3 用户查看考勤页面图

(4)用户查看工资页面。如页面图 4-4。

(5)管理员页面。如页面图 4-5。

图 4-5 管理员页面图

(6)管理录入基本工资页面。如页面图 4-6。

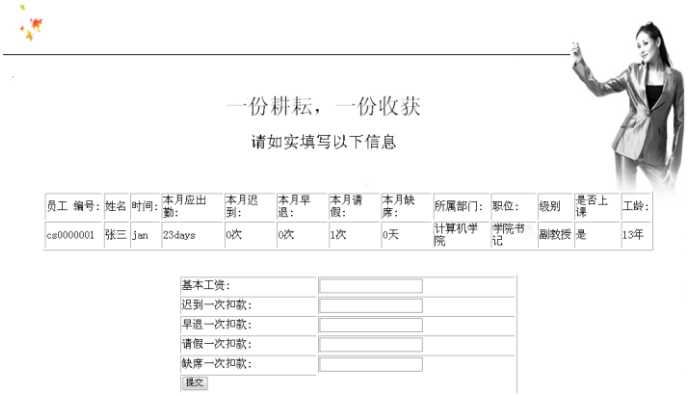

图 4-6 基本工资录入页面图

(7)录入详细工资项页面。如页面图 4-7。

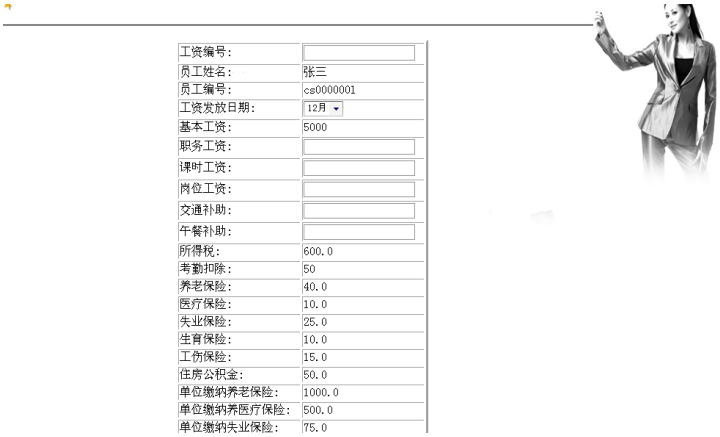

图 4-7 详细工资项录入页面图

这里就不一一列举页面图，系统经过性能的测试基本实现了课题所要求的功

能。

3.3 性能分析

性能测试：对软件的各项性能指标进行测试。

从测试的结果看，多个用户同时访问时，性能存在有一个饱和点，一旦超过这个负荷，就使性能表现出下降趋势。而且这个跟 Oracle 数据库的配置有很的关系，本系统所用的 Oracle 数据库系统所用的是专用的服务器模式，它不同于共享的服务器模式。它们的区别是：专用服务器模式就是说每次在对 Oracle 进行访问的时候，Oracle 服务器的 Listener 会得到这个访问请求，然后会为这个访问创建一个新的进程来进行服务。所以说，对于每一个客户端的访问，都会生成一个新的进程进行服务，是一种类似一对一的映射关系。这种连接模式的一个很重要的特点就是 UGA(用户全局域)是存储在 PGA(进程全局域)中的，这个特性也很好说明了当前用户的内存空间是按照进程来进行分配的。

共享服务器连接则是一种在程序编写的时候通常会用到的连接池(Pool)的概念。采用这种模式的话，在数据库的初始化的时候就会创建一批服务器连接的进程，然后把这些连接进程放入一个连接池来进行管理。初始化的池中的进程数量在数据库初始化建立的时候是可以手动设置的。在连接建立的时候，Listener 首先接受到客户端的建立连接的请求，然后 Listener 去生成一个叫做调度器(Dispatcher)的进程与客户端进行连接。调度器把客户端的请求放在 SGA(系统全局域)的一个请求队列中，然后再共享服务器连接池中查找有无空闲的连接，然后让这个空闲的服务器进行处理。处理完毕以后再把处理结果放在 SGA 的相应队列中。调度器通过查询相应队列，得到返回结果，再返回给客户端。这种连接模式的优点在于服务器进程的数量可以得到控制。考虑到我们学校的员工的数量不是很大，所以采用了专用服务器模式，这样每个连接进来的用户不会出现由于进程不够而出现的等待情况。

在性能测试中我们通常注意以下四方面数据:负载数据 、数据流量、软件本身消耗资源情况、系统使用情况。本系统是一个工资管理网站，系统性能主要体现在用户对数据进行操作的时间。系统配置如下。

硬件环境配置：INTEL(R) Core(TM) 2 DUO CPU T8100 2.10GHz,2GB 内存。

软件环境配置：Windows XP、JDK 1.5.0、ORACLE、Eclipse、Tomcat 和 Dreamweaver。

利用不同用户数点击同一页面所需的响应时间来体现系统的性能。如性能分析表 4-2 所示。

表 4-2 性能分析表

用户数	响应时间
1 人	00：00：03

2 人	00：00：05
3 人	00：00：08