南昌大学实验报告

学生姓名：丁俊　　学　　号：8003119100　专业班级：信息安全 193 班

实验类型：☐ 验证 ☐ 综合 ☐ 设计 ☐ 创新　实验日期：**5.04**　实验成绩：

一、实验项目名称

JavaBean 实验: 用户注册

二、实验目的

1. 掌握 JavaBean 属性

2. 掌握<jsp:useBean>指令、<jsp:setProperty>指令和<jsp:getProperty>指令

3. 掌握利用 Javabean 连接数据库

4. 掌握 JSP+JavaBean 的使用

5. 掌握分页显示算法

6. 掌握多条件查询和数据添加

三、实验要求

1）建立用户数据库表

2）利用 JavaBean 实现用户注册，注册页面包括学号、用户名、密码、密码确认、邮件。先使用 setProperty 设置 JavaBean 属性的值，然后使用 getProperty 在页面输出所有值。

3）利用 JavaBean 连接数据库并保存注册信息，同时验证数据库中是否存在刚输入的学号，如存在，给以提示并返回注册页面。同时统计显示已注册的人数。

4）实现分页显示，包括第一页、上一页、下一页、最后一页以及可选跳转页

5）实现多条件查询和数据添加。

四、主要仪器设备及耗材

Windows10 IntelliJ IDEA 2020.2.3 x64

五、实验步骤

1、创建学生对象

创建学生的 javabean 对象，可以用 setProperty 和 getPropety 获取和显示学生对象的属性值，学生对象包含编号 id、学号 stuId、学生姓名 stuName、学生密码 stuWord、邮件 email，另外生成了 get 和 set 方法，还有无参和有参的构造函数、toString 方法。

```
public class Student {
    public Integer id;
    private String stuId;
    private String stuName;
    private String stuWord;
    private String email;

    public String getStuId() { return stuId; }

    public void setStuId(String stuId) { this.stuId = stuId; }

    public String getStuName() { return stuName; }

    public void setStuName(String stuName) { this.stuName = stuName; }

    public String getStuWord() { return stuWord; }

    public void setStuWord(String stuWord) { this.stuWord = stuWord; }

    public String getEmail() { return email; }

    public void setEmail(String email) { this.email = email; }

    public Integer getId() { return id; }

    public void setId(Integer id) { this.id = id; }
    public Student() {

    }
    public Student(Integer id, String stuId, String stuName, String stuWord, String email) {
        this.id = id;
        this.stuId = stuId;
        this.stuName = stuName;
        this.stuWord = stuWord;
        this.email = email;
    }
```

2、创建分页对象

分页对象是处理与分页的属性相关的一些数据，把每一页封装成一个对象数据，便于以页为单位进行分页和操作，页的属性有当前页码、总页数、每页的记录数、总的记录数、当前页的学生信息集合、当前页的 url(可用于记录查询条件)链接。同样地，pageBean 对象也自动生成 get 和 set 方法以及 toString 方法。

```
public class PageBean<T> {
    public static final int PAGE_SIZE = 4;
    private int currentPage; //当前页
    private int totalPage;//总页数
    private int pageSize;//每页的记录数
    private int totalSize; //总的记录数
    private List<T> list; //当前页的学生集合
    private String url;

    public int getCurrentPage() { return currentPage; }

    public void setCurrentPage(int currentPage) { this.currentPage = currentPage; }

    public int getTotalPage() { return totalPage; }

    public void setTotalPage(int totalPage) { this.totalPage = totalPage; }

    public int getPageSize() { return pageSize; }

    public void setPageSize(int pageSize) {
        this.pageSize = pageSize;
    }
```

3、工具类

Handlestring 类

用于记录分页时的多条件查询的条件，便于获取链接到 pageBean 的 url 中，拼接字符串形成新的地址，加上如“&stuName=”，并返回这个字符串。

```
public class Handlestring {
    public static String getAddress(String stuName,String stuId,String email){
        StringBuilder address = new StringBuilder("studentServlet?action=pageByThree");
        if(!TextUtils.isEmpty(stuName)){
            address.append("&stuName=").append(stuName);
        }
        if(!TextUtils.isEmpty(stuId)){
            address.append("&stuId=").append(stuId);
        }
        if(!TextUtils.isEmpty(email)){
            address.append("&email=").append(email);
        }
        if(TextUtils.isEmpty(stuName)&&TextUtils.isEmpty(stuId) && TextUtils.isEmpty(email)){
            address.delete(26,33);
        }

        // 返回获得的字符串地址
        return address.toString();
    }
```

jdbcUtils 类

数据库连接类，利用数据库连接池连接数据库对其操作，这个类中有连接和释放操作，先读取数据库配置文件进行连接。

jdbc1.properties

```
username = root
password = 1234
url=jdbc:mysql://localhost:3306/student?useSSL=true&serverTimezone=UTC&characterEncoding=utf-8
driverClassName = com.mysql.cj.jdbc.Driver
initialSize=5
maxActive=10
```

```
public class jdbcUtils {
    private static DruidDataSource dataSource;
    static {
        try {
            Properties properties = new Properties();
            // 读取jdbc配置文件
            InputStream inputStream = JdbcUtils.class.getClassLoader().getResourceAsStream( name: "jdbc1.properties")
            // 从流中加载数据
            properties.load(inputStream);
            dataSource  = (DruidDataSource) DruidDataSourceFactory.createDataSource(properties);
//              System.out.println(dataSource.getConnection());
        }catch (Exception e){
            e.printStackTrace();
        }
    }
    public static DataSource getDataSource(){
        return dataSource;
    }
    public static Connection getConnection(){
        Connection conn = null;
        // 获取数据库池中的连接,如返回null连接失败
        try {
            conn = dataSource.getConnection();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return conn;
    }
    public static void close(Connection conn){
        if(conn != null){
            try {
                conn.close();
            }catch (SQLException e){
                e.printStackTrace();
            }
        }
    }
```

TextUtils 类

封装判断字符串是否为空的方法。

```
public class TextUtils {
    public static boolean isEmpty(CharSequence s){
        return s==null || s.length() == 0;
    }
}
```

ToInt 类

封装把字符串转化为 int 类型的数据

```
public class ToInt {
    public static int parseInt(String strInt,int defaultValue){
        try{
            return Integer.parseInt(strInt);
        }catch (Exception e){

        }
        return defaultValue;
    }
```

4、Dao 层

StudentDao 接口

```
public interface StudentDao {

    /**
     * 是否存在该学号
     * @param stuId
     * @return 返回null表示没有这个用户
     */
    public Student queryBystuId(String stuId) throws SQLException;
    /**
     * 保存学生信息到数据库
     * @param student
     * @return 返回-1 表示操作失败
     */
    public int saveStudent(Student student) throws SQLException;
    /**
     * 特定sql语句查询得到的总记录数
     *
     * @param sql
     * @return 返回一个int表示数量
     */
    public int findCount(String sql,Object... args) throws SQLException;

    /**
     * 返回特定sql语句查询获得的信息列表
     * @param sql
     * @param args
     * @return
     * @throws SQLException
     */
    public List<Student> queryForList(String sql, Object... args) throws SQLException;
}
```

StudentImpl 函数

实现 StudentDao 接口，利用 QueryRunner 对象进行数据库操作。

1、通过学号 stuId 查询单个学生

```
public Student queryBystuId(String stuId) throws SQLException {
    return runner.query( sql: "select * from class where stuId = ?",new BeanHandler<Student>(Student.class),stuId);
}
```

2、插入一个学生信息到数据库保存

```
public int saveStudent(Student student) throws SQLException {
    return runner.update( sql: "insert into class values(null , ?,?,?,?)"
            student.getStuId(),
            student.getStuName(),
            student.getStuWord(),
            student.getEmail()
    );
}
```

3、查询 sql 语句查到的记录总数量，并返回一个整数

```
public int findCount(String sql, Object... args) throws SQLException {
    Number cnt = (Number) runner.query(sql,new ScalarHandler(),args);
    return cnt.intValue();
}
```

4、返回一个学生列表

```
public List<Student> queryForList(String sql, Object... args) throws SQLException {
    return runner.query(sql,new BeanListHandler<Student>(Student.class),args);
}
```

5、service 层

StudentService 接口和 StudentserviceImpl 实现类

```java
public boolean existId(String id) throws SQLException;
/**
 * 把一个学生信息存入到数据库中
 * @param student
 * @throws SQLException
 */
public void registStudent(Student student) throws SQLException;

/**
 * 查询单个分页的学生列表
 * @param sql
 * @param args
 * @return
 * @throws SQLException
 */
public List<Student> queryForPaegItmes(String sql, Object... args) throws SQLException;

/**
 * 查询所有学生返回的页数据
 * @param pageNo
 * @param pageSize
 * @return
 */
public PageBean<Student> page(int pageNo,int pageSize) throws SQLException;

/**
 * 根据多条件组合查询得到分页的学生列表
 * @param pageNo
 * @param pageSize
 * @param stuName
 * @param stuId
 * @param email
 * @return
 * @throws SQLException
 */
public PageBean<Student> searchStudent(int pageNo, int pageSize, String stuName, String stuId,String email) throws SQLException;
```

1、 判断一个学号是否存在

```java
public boolean existId(String id) throws SQLException {
    if(studentDao.queryBystuId(id)==null){
        return false;
    }
    return true;
}
```

在数据库中根据学号查询，如果查不到表示不存在返回 false；查到了表示存在返回 true。

2、 注册学生保存

```java
public void registStudent(Student student) throws SQLException {
    studentDao.saveStudent(student);
}
```

注册学生即把该学生信息插入到数据库中。

3、 返回一个学生记录的列表

```java
public List<Student> queryForPaegItmes(String sql, Object... args) throws SQLException {
    return studentDao.queryForList(sql,args);
}
```

直接调用 studentDao 中的查询列表函数，返回多个学生记录。

4、 无条件的查询分页函数(查询所有)

```
public PageBean<Student> page(int pageNo, int pageSize) throws SQLException {
    PageBean<Student> page = new PageBean<~>();

    page.setPageSize(pageSize);
    // 总记录数
    int totalSize = studentDao.findCount( sql: "select count(*) from class");
    // 设置总记录数
    page.setTotalSize(totalSize);
    // 总页码
    int totalPage = (totalSize % pageSize > 0) ?  totalSize/pageSize+1 :  totalSize/pageSize;
    page.setTotalPage(totalPage);

    if(pageNo < 1){
        pageNo = 1;
    }
    if(pageNo > totalPage){
        pageNo = totalPage;
    }
    // 设置当前页码
    page.setCurrentPage(pageNo);
    // 查询起点
    int begin = (page.getCurrentPage()-1)*pageSize;
    if(begin < 0){
        begin = 0;
    }
    String sql = "select * from class" + " limit " + begin + "," + pageSize;
    List<Student> items = studentDao.queryForList(sql);
    page.setList(items);
    return page;
}
```

先查询所有数量，然后根据总页码=总记录数/每页记录数(除不尽加 1)，查询起点 begin=(当前页码-1)*每页记录数，然后利用 limit 子句限制查询范围即可获得对应页码的数据，并保存在 page 对象中的 list 列表中。

5、 多组合条件查询分页函数

多组合条件查询函数需要判断参数是否存在，存在即保存在一个 list 集合中，后作为参数传给封装好的 sql 查询函数。Cntsql 用来查询总数量，sql 用来查询并返回接收数据。

```
String sql = "select * from class where 1=1  ";
// 这个是查找数量的语句
String cntSql = "select count(*) from class where 1=1";
List<String> list = new ArrayList<~>();
// 生成sql语句
if(!TextUtils.isEmpty(stuName)){
    sql = sql + " and stuName like ? ";
    cntSql = cntSql + " and stuName like ? ";
    list.add("%" + stuName + "%");
}
if(!TextUtils.isEmpty(stuId)){
    sql = sql + " and stuId > ? ";
    cntSql = cntSql + " and stuId > ? ";
    list.add(stuId);
}
if(!TextUtils.isEmpty(email)){
    sql = sql + " and email = ? ";
    cntSql = cntSql + " and email= ? ";
    list.add(email);
}
// 总记录数
int TotalCount = studentDao.findCount(cntSql,list.toArray());
```

总页码和起点和上面无条件查询是一样的，只要知道总记录数、每页大小和当前页码即可。

```
pageBean.setTotalPage(pageTotal);
if(pageNo < 1){
    pageNo = 1;
}
if(pageNo > pageTotal){
    pageNo = pageTotal;
}
// 设置当前页码
pageBean.setCurrentPage(pageNo);
int begin = (pageBean.getCurrentPage()-1)*pageSize; // 查询起点\
// 有可能begin计算会小于0
if(begin < 0){
    begin = 0;
}
sql = sql + " limit "+ begin + "," + pageSize; // 先设置limit值，这样就不用传了
List<Student> items = queryForPaegItmes(sql,list.toArray());
pageBean.setList(items);
return pageBean;
```

注意需要判断 begin 的正负，因为有可能查询到的总数量为 0，从而导致 begin 查询起始值小于 0，所以要避免 begin 小于 0，从而发生异常。

6、web-servlet 层

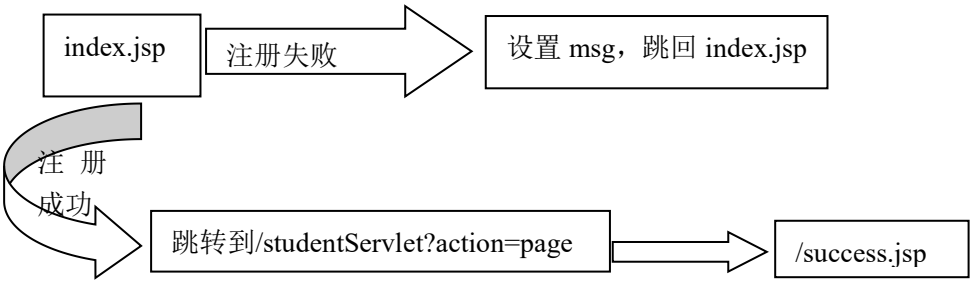
web.xml 配置

```
<servlet>
    <servlet-name>TestServlet</servlet-name>
    <servlet-class>TestServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>TestServlet</servlet-name>
    <url-pattern>/test</url-pattern>
</servlet-mapping>

<servlet>
    <servlet-name>StudentServlet</servlet-name>
    <servlet-class>StudentServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>StudentServlet</servlet-name>
    <url-pattern>/studentServlet</url-pattern>
</servlet-mapping>
```

注册功能 TestServlet

```
public class TestServlet extends HttpServlet {
    private StudentService studentService = new StudentServiceImpl();
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletExcep
        req.setCharacterEncoding("utf-8");
        String stuId = req.getParameter( s: "stuId");
        String stuName = req.getParameter( s: "stuName");
        String stuWord = req.getParameter( s: "stuWord");
        String confWord = req.getParameter( s: "confWord");
        String email = req.getParameter( s: "email");

        try {
            if(studentService.existId(stuId)){
                req.setAttribute( s: "msg", o: "学生ID已存在");
                req.setAttribute( s: "stuName",stuName);
                req.setAttribute( s: "email",email);
                req.getRequestDispatcher( s: "index.jsp").forward(req,resp);
            }else if(!stuWord.equals(confWord)){ // 密码不一致
                req.setAttribute( s: "msg", o: "两次密码不一致");
                req.setAttribute( s: "stuId",stuId);
                req.setAttribute( s: "stuName",stuName);
                req.setAttribute( s: "email",email);
                req.getRequestDispatcher( s: "index.jsp").forward(req,resp);
            }else if(!studentService.existId(stuId) && stuWord.equals(confWord)){
                // 注册成功存入数据库
                studentService.registStudent(new Student( id: null,stuId,stuName,stuWord,email));
                resp.sendRedirect( s: req.getContextPath() + "/studentServlet?action=page");
            }
        } catch (Exception e) {
```

重构 doPost 方法，通过 req.getParameter 方法获取参数并判断学生 id 是否在数据库中已经存在，如果不存在设置错误信息 msg 待跳回注册页面后通过 request 域中的数据接收并显示；反之跳转到分页界面。



StudentServlet

通过 method 反射方法获取字符串调用对应方法

```
protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    req.setCharacterEncoding("utf-8");
    String act = req.getParameter( s: "action");

    try {
        // 获得action字符串获取相应的业务 方法反射对象
        Method method = this.getClass().getDeclaredMethod(act,HttpServletRequest.class,HttpServletResponse.class);
        // 调用目标业务方法
        method.invoke( obj: this,req,resp);
    }catch (Exception e){
        e.printStackTrace();
    }finally {

    }
```

多条件分页方法

```
public void pageByThree(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IO
    int currentPage = ToInt.parseInt(req.getParameter( s: "pageNo"), defaultValue: 1);
    int pageSize = PageBean.PAGE_SIZE;

    // 有可能为空值的
    String stuName = req.getParameter( s: "stuName");
    String stuId = req.getParameter( s: "stuId");
    String email = req.getParameter( s: "email");

    PageBean<Student> page = studentService.searchStudent(currentPage,pageSize,stuName,stuId,email);
    // 设置url地址
    page.setUrl(Handlestring.getAddress(stuName,stuId,email));
    req.setAttribute( s: "page",page);
    req.getRequestDispatcher( s: "/success.jsp").forward(req,resp);
```

调用 sudentService 封装的 searchStudent 方法，传入当前所需页码、每页大小、学生姓名、学号、邮件进行处理返回的是搜索分页的结果，并根据传入的参数设置 url 地址保存信息且可以在 index.jsp 中进行回显，这样每次查询的信息就都是此时选择的组合条件。

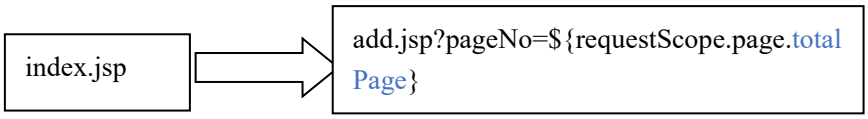
```
 <td>学号：</td>
 <td><input type="text" id="stuId" name="stuId" value=${requestScope.stuId}> </td>
/tr>
tr>
 <td>用户名:</td>
 <td><input type="text" id="stuName" name = "stuName" value="${requestScope.stuName}"></td>
/tr>
tr>
 <td>密码：</td>
 <td><input type="text" id="stuWord" name="stuWord"></td>
/tr>
tr>
 <td>确认密码:</td>
 <td><input type="text" id="confWord" name = "confWord"></td>
/tr>
tr>
 <td>邮件:</td>
 <td><input type="email" id="email" name = "email" value="${requestScope.email}"></td> <!--email输入框-->
```

6、结果 javabean 对象显示

```
<tbody>
<c:forEach items = "${requestScope.page.list}" var="stu">
    <jsp:setProperty name="Student" property="stuId" value="${stu.stuId}"/>
    <jsp:setProperty name="Student" property="stuName" value="${stu.stuName}"/>
    <jsp:setProperty name="Student" property="stuWord" value="${stu.stuWord}"/>
    <jsp:setProperty name="Student" property="email" value="${stu.email}"/>

  <tr>
      <td><jsp:getProperty name="Student" property="stuId"/></td>
      <td><jsp:getProperty name="Student" property="stuName"/></td>
      <td><jsp:getProperty name="Student" property="stuWord"/></td>
      <td><jsp:getProperty name="Student" property="email"/></td>
  </tr>

</c:forEach>
```

利用 c:forEach 循环 request 域中 page 对象中的 list 集合，通过 setProperty 和 getProperty 获取和显示数据。

7、添加学生



添加学生就显示最后一页 totalPage 的数据。

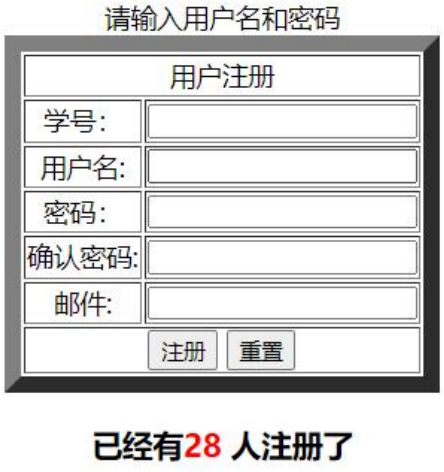
8、分页条的显示

```
<div class="page-nav" align="center">
    <%--            大于首页才显示--%>
    <c:if test="${requestScope.page.currentPage > 1}">
        <a href = "${requestScope.page.url}&pageNo=1">首页</a>
        <a href="${requestScope.page.url}&pageNo=${requestScope.page.currentPage-1}">上一页</a>
    </c:if>
    <c:if test="${requestScope.page.currentPage <= 1}">
        首页   上一页
    </c:if>
    第【${requestScope.page.currentPage }】页
    <%--    防止末页越界--%>
    <c:if test="${requestScope.page.currentPage < requestScope.page.totalPage}">
        <a href="${requestScope.page.url}&pageNo=${requestScope.page.currentPage+1}">下一页</a>
        <a href="${requestScope.page.url}&pageNo=${requestScope.page.totalPage}">末页</a>
    </c:if>
    <c:if test="${requestScope.page.currentPage >= requestScope.page.totalPage}">
        下一页   末页
    </c:if>
    <%--            param.pageNo显示当前页码--%>
    共[${requestScope.page.totalPage}]页 [${requestScope.page.totalSize}]条记录 转到第
    <input value="${param.pageNo}" name="pn" id="pn_input" style="..."/>页
    <input id="Topage" type="button" value="确定">
```

```
<script type="text/javascript">
    $(function () {
        // 跳到指定的页码
        $("#Topage").click(function () {
            var pageNo = $("#pn_input").val();

            var pageTotal = ${requestScope.page.totalPage}; // 获取总页码
            if(pageNo<1 || pageNo > pageTotal){
                alert("不能跳转,页面超过范围或不存在");
                return;
            }
            window.location.href = "${pageScope.basePath}${requestScope.page.url}&pageNo="+
                pageNo;
        });
    });
</script>
div>
```

六、实验数据及处理结果

主界面



重复注册 id

注册成功

恭喜注册成功

条件查询

姓名模糊查询

用户名包含“俊”且学号大于 8010。

三个条件

邮件查询

添加学生

成功显示在最后一页

七、思考讨论题或体会或对改进实验的建议