一、 题型

题目 编号	小题 数目	题型	分值
一	3	程序填空题	30
二	4	阅读程序结果	20
三	3	编程题	30
四	1	界面题	20

二、 重点关注

1. 掌握 sf.jsp 中的十个算法，要求能举一反三。

```
//累加
int sum(int n)
{
    int sum1 = 0;
    for (int i = 1; i <= n; i++)
        sum1 += i;
    return sum1;
}

int fun12(int n)
{
    int result = 0;
    int i=1;

}

//阶乘
int fun1(int n)
{
    int result = 1;
    for (int i = 1; i <= n; i++)
        result *= i;
    return result;
}

//素数
boolean fun2(int x)
{
    boolean flag = true;
    for (int i = 2; i <= StrictMath.sqrt(x);
i++)
    {
        if (x % i == 0)
        {
            flag = false;
            break;
        }
    }
    return flag;
}

//闰年
boolean fun3(int year)
{
    if (year % 4 == 0 && year % 100 != 0 || year %
400 == 0)
    {
        return true;
    }
    else
    {
        return false;
    }
}

//整数倒
int fun4(int x)
{
    int result = 0;
    while (x > 0)
    {
        int yushu = x % 10;
        result = result * 10 + yushu;
        x = x / 10;
    }
    return result;
}
```

```
//是否是回文
boolean fun5(String str)
{
    boolean result = true;
    int i = 0;
    int j = str.length() - 1;
    while (i < j)
    {
        if (str.charAt(i) == str.charAt(j))
        {
            i++;
            j--;
        }
        else
        {
            result = false;
            break;
        }
    }
    return result;
}


//求最大数
int fun6(int[] a)
{
    int max = a[0];
    int n = a.length;
    for (int i = 1; i < n; i++)
    {
        if (a[i] > max)
            max = a[i];
    }
    return max;
}


//从小到大排序
void fun7(int[] a)
{
    int min;
    int n = a.length;
    for (int i = 0; i < n - 1; i++)
    {
        min = i;
        for (int j = i + 1; j < n; j++)
        {
            if (a[j] < a[min])
            {
        min = j;
            }
        }
        int tmp = a[i];
        a[i] = a[min];
        a[min] = tmp;
    }
}


//将数组中 x 的倍数变为 0
void fun8(int[] a,int x)
{
    int n = a.length;
    for (int i = 0; i < n; i++)
    {
        if (a[i] % x == 0)
        {
            a[i] = 0;
        }
    }
}


//求最大公约数
int fun9(int m, int n)
{
    int r;

    do
    {
        r = m % n;
        if (r != 0)
        {
            m = n;
            n = r;
        }
    } while (r != 0);

    return n;
}
```

2. 要求能对单表进行增、删、改、查。（JDBC 数据库操作）

1） 驱动的选择

```
// 1.定义并声明常用字段
private static final String JDBC_DRIVER = "驱动名";
private static String url = "数据库连接串URL";
private static String user = "root";
private static String pwd = "password";
```

注：常见数据库驱动、默认端口号、URL、账户名如下

数据库	驱动名称	端口	URL	账户
MySQL	com.mysql.jdbc.Driver	3306	jdbc:mysql://localhost:端口号/数据库名	root
MariaDB	org.mariadb.jdbc.Driver	3306	jdbc:mysql://localhost:端口号/数据库名	root
SQL Server	com.microsoft.sqlserver.jdbc.SQLServerDriver	1433	jdbc:microsoft:sqlserver://localhost:端口号;DatabaseName=数据库名	sa
Oracle	oracle.jdbc.driver.OracleDriver	1521	jdbc:oracle:thin:@localhost:端口名:orcl	sys

```
// 2.定义并声明SQL操作对象
private static Connection conn = null; //数据库连接对象
private static Statement st = null;    //状态对象
private static ResultSet rs = null;    //结果集对象
```

注：以上均为类内成员变量声明，如在方法（函数）内声明则去掉“**private static final**”等修饰符。

2） 创建连接

```
//方法1：获取数据库连接
Class.forName(JDBC_DRIVER);                          //1、注册驱动
conn = DriverManager.getConnection(url, user, pwd);  //2、获取连接
//方法2：获取数据库连接(通过DBCP数据库连接池)
Context ctx = new InitialContext();
DataSource ds=(DataSource) ctx.lookup("java:comp/env/jdbc/DBPool");
conn=ds.getConnection();
```

3） 创建 statement

```
//类型 1：创建 statement
conn.setAutoCommit(false);        //关闭自动事务
st = conn.createStatement();      //创建 statement
//类型 2：创建 prepareStatement
PreparedStatement ps;             //声明 preparestatement
String sql="SQL 语句";            //准备 SQL 语句，如 insert into lover values(?,?,?)
ps = (PreparedStatement) conn.prepareStatement(sql); //创建 preparestatement
```

4） 执行 SQL

```
//类型 1：使用 Statement
String sql="SQL 语句";            //准备 SQL 语句
st.execute(sql);                  //执行 SQL 语句
conn.commit();                    //提交事务
//类型 2：使用 prepareStatement，需要先填充准备 SQL 语句中的占位符
```

```
ps.setInt(1,21);//代表设置给第一个?号位置的值为 Int 类型的 21
ps.setString(2,"suwu150");//代表设置给第二个?号位置的值为 String 类型的 suwu150
java.util.Date utilDate=new java.util.Date();//类型转换，由 util 类型的 date 转化为 sql 类型的
ps.setDate(3, new java.sql.Date(utilDate.getTime()));
ps.execute();                    //执行 prepareStatement
```

封装后的数据库操作相关方法如下：

```
/**
 * @ 函数名称：executeBatch
 * @ 功能描述：根据查询 SQL 语句进行增删改操作。
 * @ 传入参数：用于查询的 SQL 语句 list (ArrayList<HashMap<String,Object>>)
 * @ 返回类型：boolean
**/
public static boolean executeBatch(ArrayList<String> list) {
    boolean flag = true;// 返回值默认为 true
    try {
        conn = getConn();// 调用 getConn()方法，初始化数据库连接
        conn.setAutoCommit(false);
        st = conn.createStatement();
        for (int i = 0; i < list.size(); i++) {
            st.addBatch(list.get(i));
        }
        st.executeBatch();
        conn.commit();// 执行事务
        conn.setAutoCommit(true);

    } catch (Exception ex) {
        try {
            conn.rollback();// 事务回滚
        } catch (SQLException e) {
            e.printStackTrace();
        }
        flag = false;// 执行失败，返回 false
        ex.printStackTrace();
    } finally {
        finallyHandle(conn, st, rs);// 关闭数据库连接
    }
    return flag;
}

/**
 * @ 函数名称: getDataSetInfoByCon
 * @ 功能描述：根据查询 SQL 语句、页码及页数返回部分多条记录。
 * @ 传入参数：用于查询的 SQL 语句、页码、页数
 * @ 返回类型： (ArrayList<HashMap<String,Object>>)
*/
public static ArrayList<HashMap<String, String>> getDataSetInfoByCon(String sql, int rowCount,
int page) {
    Connection conn = null;
```

```
    ArrayList<HashMap<String, String>> result = null;
    Statement st = null;
    ResultSet rs = null;
    ResultSetMetaData rsmd = null;
    try {
        conn = getConn();
     st = conn.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_READ_ONLY);
        if (rowCount > 0)
            st.setMaxRows(page * rowCount);
        rs = st.executeQuery(sql);
        if (page >= 0 && rowCount > 0)
            rs.absolute((page - 1) * rowCount);
        rsmd = rs.getMetaData();
        result = new ArrayList<HashMap<String, String>>();
        while (rs.next()) {
            int columnCount = rsmd.getColumnCount();
            HashMap<String, String> record = new HashMap<String, String>();
            for (int i = 1; i <= columnCount; i++) {
                record.put(rsmd.getColumnName(i), rs.getString(i));
            }
            result.add(record);
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        finallyHandle(conn, st, rs);
    }
    return result;
}
/**
 * @ 函数名称: getRowCount
 * @ 功能描述：根据查询 SQL 语句返回记录行数。
 * @ 传入参数：用于查询的 SQL 语句
 * @ 返回类型：int
*/
public static int getRowCount(String sql) {
    Connection conn = null;
    Statement st = null;
    ResultSet rs = null;
    int length = 0;
    try {
        conn = getConn();
        st = conn.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_READ_ONLY);
        rs = st.executeQuery(sql);
        rs.last();
        length = rs.getRow();
    } catch (Exception e) {
        e.printStackTrace();
```

```
    } finally {
        finallyHandle(conn, st, rs);
    }
    return length;
}

/**
 * @ 函数名称: getDataCount
 * @ 功能描述：获取行数
 * @ 传入参数：用于查询的参数与表名
 * @ 返回类型：int
 * @ 文件作者：DukeWF
 * @ 创建时间：2018-05-29
 * @ 版本编号：1.00
 **/
public static int getDataCount(String tablename, String key, String value) {
    int rowCount = 0;
    try {
        String sql = "SELECT COUNT(*) AS record_ FROM " + tablename + " WHERE "+ key +" = ?";
        System.out.println(sql);
        conn = getConn();

        PreparedStatement prestmt;
        prestmt = conn.prepareStatement(sql);
        prestmt.setString(1,value);

        rs = prestmt.executeQuery();
        if (rs.next()) {
            rowCount = rs.getInt("record_");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        finallyHandle(conn, st, rs);
    }
    System.out.print(rowCount);
    return rowCount;

}
```

5） 结果集的遍历

```
/**
 * @ 函数名称：convertList
 * @ 功能描述：将结果集遍历至 List 中
 * @ 传入参数：查询结果集 rs
 * @ 返回类型：List
**/
public static List convertList(ResultSet rs) throws SQLException {
    List list = new ArrayList();
    ResultSetMetaData md = rs.getMetaData();//获取键名
    int columnCount = md.getColumnCount();//获取行的数量
    while (rs.next())
    {
        Map rowData = new HashMap();//声明 Map
        for (int i = 1; i <= columnCount; i++)
        {
            rowData.put(md.getColumnName(i), rs.getObject(i));//获取键名及值
        }
        list.add(rowData);
    }
    return list;
}
```

6） 内容输出

7） 数据库结束后收尾工作（分别关闭结果集、状态、连接）

```
/**
 * @ 函数名称：finallyHandle
 * @ 功能描述：对数据库操作结束进行收尾工作。
 * @ 传入参数：当前连接 conn、状态 st、结果集 rs
 * @ 返回类型：void
**/
private static void finallyHandle(Connection conn, Statement st, ResultSet rs) {
    try {
        if (rs != null) {
            rs.close();
            rs = null;
        }
        if (st != null) {
            st.close();
            st = null;
        }
        if (conn != null) {
            conn.close();
            conn = null;
        }
    } catch (Exception ex) {
        ex.printStackTrace();
```

```
    }
}
```

3. 掌握 EasyUI 中的 combobox、datagrid 控件的数据展示，能从数据库中读取数据展现在 combobox 或 datagrid 中。

ComboBox

前端：

```
<input class="easyui-combobox" name="politicalstate" id="politicalstate" />
```

JS：

```
$(function() {
    $("#politicalstate").combobox({
        url : "SystemStudentService?op=politicalstate",
        valueField : "politicalstate_id",
        textField : "politicalstate_name",
        panelHeight : 'auto'
    });
})
```

Datagrid

前端：

```
<table id="info" class="easyui-datagrid" width="100%"
    style="height:100%;" border="0" cellpadding="0" cellspacing="0"
    data-options=" toolbar:'#tb'">
    <thead>
        <tr>
            <th data-options="field:'sno',width:120,align:'center'">学 号</th>
            <th data-options="field:'sname',width:120,align:'center'">姓 名</th>
            <th data-options="field:'age',width:100,align:'center'">年 龄</th>
            <th data-options="field:'politicalstate',width:120,align:'center'">政治面貌</th>
            <th data-options="field:'birthday',width:120,align:'center'">出生日期</th>
            <th data-options="field:'address',width:250,align:'center'">地址</th>
        <th data-options="field:'phone',width:100,align:'center',hidden:'true'">联系方式</th>
            <th data-options="field:'institute',width:120,align:'center'">学院</th>
            <th data-options="field:'demo',width:180,align:'center'">备 注</th>
        </tr>
    </thead>
</table>
```

JS：

```
    $("#info").datagrid({
        loadMsg : "数据加载中，请等待...",
```

```
        iconCls : 'icon-issue',
        nowrap : false,
        striped : true,
        collapsible : true,
        rownumbers : true,
        pagination : true,
        singleSelect : true,
        autoRowHeight : true,
        fitColumns : false,
        pageSize : 10,
        pageList : [ 10, 20, 30, 40 ],
        cache : false,
        url : "SystemStudentService?op=init"
    });
```

后端：

```
//Servlet：SystemStudentService
@WebServlet("/SystemStudentService")

public class SystemStudentService extends HttpServlet {
  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
  ServletException, IOException {
    // TODO Auto-generated method stub
    String op = request.getParameter("op");
    switch (op) {
    case "politicalstate":
        getComboBox(response, "politicalstate");
        break;

    case "init":
        String row = request.getParameter("rows");
        String page = request.getParameter("page");
        getStudentInfo(response, "", page, row);
        break;

    default:
        break;
    }
  }

  private void getComboBox(HttpServletResponse response, String type) {
    String result = "";
    ArrayList<HashMap<String, String>> dt = null;
    String sql;
    try {
        sql = "SELECT * FROM " + type;
        dt = DBUtil.getDataSet(sql);
        result = JSON.toJSONString(dt);
```

```
        System.out.println(result);
        response.setCharacterEncoding("utf-8");
        PrintWriter out = response.getWriter();

        out.print(result);
        out.close();
    } catch (Exception ex) {
        ex.printStackTrace();
    }
}

private static String getStudentInfo(HttpServletResponse response, String con, String page, String
row) {
        String result = "";
        Map<String, Object> map = new HashMap<String, Object>();
        ArrayList<HashMap<String, String>> dt = null;
        String sql;

        int rowscount = 0;
        if (con == null)
            con = "";
        if (row == null)
            row = "0";
        if (page == null)
            page = "0";
        try {
            int r = Integer.parseInt(row);
            int p = Integer.parseInt(page);
            if (!con.equals("")) {
                sql = "select * from student where " + con;
            } else {
                sql = "select * from student";
            }
            dt = DBUtil.getDataSetInfoByCon(sql, r, p);
            rowscount = DBUtil.getRowCount(sql);
            map.put("total", rowscount);
            map.put("rows", dt);
            result = JSON.toJSONString(map);
            response.setCharacterEncoding("utf-8");
            PrintWriter out = response.getWriter();

            out.print(result);
            out.close();
        } catch (Exception ex) {
            ex.printStackTrace();
        }
        return result;
    }
}
```

4. 掌握 HTML 中的常用标记，表格、超链接、表单等。

5. 掌握实际项目中的常见功能：登录实现（包括简单的界面）、具体功能模块的操作。

6. 掌握 Session、Cookie 的使用和操作。

Session 相关操作

```
HttpSession session = request.getSession();//Servlet 中实例化 session 对象

session.setAttribute(K,V);//存入数据

V=session.getAttribute(K);//取出数据

session.invalidate();//手动销毁 session
```

Cookie 相关属性

name：Cookie 的名称；
value：Cookie 的值；
comment：Cookie 的注释；
domain：可以看到 Cookie 的域；
maxAge：Cookie 的失效时间；正值表示 Cookie 会在指定的时间后过期，负值表示浏览器关闭的时候过期，0 会导致 Cookie 被删除；
path：可以看到 Cookie 的 URL；
secure：是否需要使用安全连接来传输；
version：版本；
isHttpOnly：HttpOnly 的 Cookie 将不会暴露给客户端的脚本代码；
PS：需要注意的是，Cookie 的名称要符合标识符的命名规则，同时不允许为【Comment，Discard，Domain，Expires，Max-Age，Path，Secure，Version】这几个关键字，也不允许以**“$”**开头。

Cookie 的增删改查

```
//1.Cookie 创建后通过 HttpServletResponse 添加。
public static void addCookie(HttpServletResponse response, String name, String value, int maxAge)
{
        Cookie cookie = new Cookie(name, value);
        cookie.setPath("/");
        if (maxAge > 0) {
            cookie.setMaxAge(maxAge);
        }
        response.addCookie(cookie);
}
//2.Cookie 通过 HttpServletRequest 获取，如下获取全部 Cookie 并以 Map 形式存储。
private static Map<String, Cookie> readCookieMap(HttpServletRequest request) {
    Map<String, Cookie> cookieMap = new HashMap<>();
```

```
    Cookie[] cookies = request.getCookies();
    if (cookies != null) {
        for (Cookie cookie : cookies) {
            cookieMap.put(cookie.getName(), cookie);
        }
    }
    return cookieMap;
}
//3.删除 Cookie 的时候将 Cookie 的 MaxAge 置为 0 后重新添加到 HttpServletResponse 即可。
public static void deleteCookie(HttpServletRequest request, HttpServletResponse response,
String name) {
    Map<String, Cookie> cookieMap = readCookieMap(request);
    if (cookieMap.containsKey(name)) {
        Cookie cookie = cookieMap.get(name);
        cookie.setMaxAge(0);
        response.addCookie(cookie);
    }
}
```

实例：利用 Cookie 保存用户基本信息

```
//添加缓存
public String add_cookie(User user, HttpServletResponse response) throws UnsupportedEncoding
Exception {
    String username = user.getUserName();
    String userPassword = user.getUserPassword();

    //将用户名存入 cookie 并且设置 cookie 存在时长
    Cookie cookie_username = new Cookie("username", URLEncoder.encode(username,"utf-8"));
    cookie_username.setMaxAge(60*60*60);
    response.addCookie(cookie_username);

    //将密码存入 cookie 并且设置 cookie 存在时长
    Cookie cookie_userPassword = new Cookie("userPassword",URLEncoder.encode(userPassword,
    "utf-8"));
    cookie_userPassword.setMaxAge(60*60*60);
    response.addCookie(cookie_userPassword);

    return null;
}
//删除缓存
@RequestMapping("/del_cookie")
@ResponseBody
public String del_cookie(HttpServletRequest request,HttpServletResponse response){
    Cookie[] cookies = request.getCookies();
    if (cookies != null && cookies.length > 0) {
        for (Cookie cookie : cookies) {
            // 找到需要删除的 Cookie
            if("username".equals(cookie.getName())){
```

```
                // 设置生存期为 0
                cookie.setMaxAge(0);
                    // 设回 Response 中生效
                response.addCookie(cookie);
            }
            if("userPassword".equals(cookie.getName())){
                // 设置生存期为 0
                cookie.setMaxAge(0);
                // 设回 Response 中生效
                response.addCookie(cookie);
            }
        }
    }
    return null;
}
```

7. 掌握 JavaBean 的规范及编写。

1.有包
2.有默认构造器
3.实现序列化接口Serializable
注意：set/get方法可以没有
Java中的实体类要满足该规范，并且在写实体类时有如下几点建议：
1.尽量使用封装类型,因为它笔基本类型多了null,尤其数据库中可以使用null,另外基本类型的默认值为0，包装类型的默认值为null
2.使用java.sql包下的日期,因为JDBC支持这样的日期类型
以员工Emp实体类，代码如下：

```
package entity;

import java.io.Serializable;
import java.sql.Date;

public class Emp implements Serializable {
    private static final long serialVersionUID = 1L;

    private Integer empno;
    private String ename;
    private String job;
    private Integer mgr;
    private Date hiredate;
    private Double sal;
    private Double comm;
    private Integer deptno;
    public Emp(){}
    public Integer getEmpno() {
        return empno;
    }
    …
    public void setEmpno(Integer empno) {
        this.empno = empno;
```

```
    }
    …
    @Override
    public String toString() {
        return "Emp [empno=" + empno + ", ename=" + ename + ", job=" + job + ", mgr=" + mgr + ",
hiredate=" + hiredate
                + ", sal=" + sal + ", comm=" + comm + ", deptno=" + deptno + "]";
    }

}
```