**实验报告**

**学院（系）名称：**计算机科学与工程学院

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **姓名** | | 王帆 | | **学号** | 20152180 | | **专业** | 计算机科学与技术 |
| **班级** | | 2015级1班 | | **实验**  **项目** | 实验4：Web应用综合设计 | | | |
| **课程名称** | | | | Web应用程序设计与开发 | | | **课程**  **代码** | 0668026 |
| **实验时间** | | | | 2018年6月1日 1、2节  2018年6月5日 中午5、6节  2018年6月8日 1、2节  2018年6月12日 5、6节 | | | **实验**  **地点** | 7-215 |
| 考核标 准 | 实验准备（实验目的/工具熟悉情况）10分 | | 实验过程（实验方案可行性及步骤完整性）40分 | 实验报告(实验内容丰富度与格式清晰度) 30分 | | 实验结果（结论正确性以及分析合理性）20分 | **成绩** |  |
|  |  | |  |  | |  | **教师签字：** | |
| 考核内容 | 评价实验目的是否明确，实验工具是否清晰了解以及熟悉情况 | | ○可行，完整  ○可行，不完整  ○不可行，不完整 | ○丰富，清晰  ○较丰富，较清晰  ○丰富，不清晰  ○不丰富，不清晰 | | ○结论正确，分析合理  ○结论正确，分析不充分  ○结论不正确，分析不合理 |
| **一、实验目的**  （1）综合利用JDBC、JSP、Servlet、EasyUI及jQuery等技术实现一个小型Web应用系统；  （2）系统应具有用户登录功能；  （3）实现登录后的主界面；  （4）至少实现2个具体的业务相关的功能模块（包括对表的增删改查基本操作）；  （5）系统应能够对登录用户的会话的跟踪（可利用Session方式）。  **二、实验环境**  Windows操作系统，Tomcat，MyEclipse，Dreamweaver，记事本。  **三、 实验要求**  1、构建基于SQL Server2008R数据库相关的表。要求：  （1）详细列写说明各个表的结构；  （2）指出并说明各个表的作用。  2、构建系统对数据库的通用访问类，具体要求：  （1）类的实现上应具有执行查询SQL语句的返回结果集的方法；  （2）能够实现执行insert into、update及delete方法；  （3）类应进行相关资源的释放。  3、实现系统的登录功能，要求：  （1）实现用户名和密码到数据库表中的验证；  （2）用户名和密码错误的提示。  4、主界面的设计及实现，要求：  （1）应显示出当前登录用户的信息；  （2）具有打开某个功能模块的链接。  5、具体模块的功能实现，要求：  （1）设计和模块相应的数据库表；  （2）实现对模块对应表的增删改查操作。  6、选做内容：  实现统计报表功能：可利用poi技术将数据导入EXCEL文件方式实现。  **四、实验过程记录（源程序、测试用例、测试结果及心得体会等）**  1、构建基于SQL Server2008R数据库相关的表。  由于我主要使用的数据库是MariaDB，因此我将使用HendiSQL与Navcat作为数据库管理软件，对本实验数据库进行管理，并形成数据库E-R图。    图1 数据库E-R图  表1 数据表结构（用户信息）   |  |  | | --- | --- | | 用户登录注册信息管理 | | | 名称 | 数据类型 | | userid | INT | | username | VARCHAR | | password | VARCHAR |   表2 数据表结构（学生信息）   |  |  | | --- | --- | | 学生详情信息管理 | | | 名称 | 数据类型 | | sno | INT | | sname | VARCHAR | | age | VARCHAR | | politicalstate | VARCHAR | | birthday | VARCHAR | | address | VARCHAR | | phone | VARCHAR | | institute | VARCHAR | | demo | VARCHAR |   表3 数据表结构（学生简要信息）   |  |  | | --- | --- | | 学生简要信息管理 | | | 名称 | 数据类型 | | sno | INT | | sname | VARCHAR |   表4 数据表结构（政治面貌）   |  |  | | --- | --- | | 政治面貌类型管理 | | | 名称 | 数据类型 | | politicalstate\_id | INT | | politicalstate\_name | VARCHAR |   表5 数据表结构（院系）   |  |  | | --- | --- | | 院系信息管理 | | | 名称 | 数据类型 | | institute\_id | INT | | institute\_name | VARCHAR |   2、构建系统对数据库的通用访问类。  数据库连接池配置：  （1）在项目：【WebRoot】-【META-INF】下：Context.xml文件中加入如下内容：  <Context>  <Resource name="jdbc/DBPool" auth="Container"  type="javax.sql.DataSource"  factory="org.apache.commons.dbcp2.BasicDataSourceFactory"  username="用户名"  password="密码"  driverClassName="数据库驱动名"  url="数据库连接串"  maxTotal="100"  maxIdle="1000"  maxWaitMillis="5000" />  </Context>  （2）在项目：【WebRoot】-【WEB-INF】下: web.xml文件中加入如下内容：  <resource-ref>  <description>DB Connection</description>  <res-ref-name>jdbc/DBPool</res-ref-name>  <res-type>javax.sql.DataSource</res-type>  <res-auth>Container</res-auth>  </resource-ref>  数据库工具类（DBUtil.class）  /\*\*  \* @ 函数名称: getConn  \* @ 功能描述：获取数据库连接(通过连接池）  \* @ 传入参数：无  \* @ 返回类型：Connection  \* @ 文件作者：DukeWF  \* @ 创建时间：2018-04-30  \* @ 版本编号：1.00  \*\*/  **public** **static** Connection getConn(){  **try**{  Context ctx = **new** InitialContext();  DataSource ds=(DataSource) ctx.lookup("java:comp/env/jdbc/DBPool");  conn=ds.getConnection();  }**catch**(Exception e){  e.printStackTrace();  }  **return** conn;  }  /\*\*  \* @ 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sql  \* @ 返回类型：boolean  \* @ 文件作者：DukeWF  \* @ 创建时间：2018-04-30  \* @ 版本编号：1.00  \*\*/  **public** **static** **boolean** executeBatch(String sql) {  **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();  st.addBatch(sql);  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：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  \* @ 文件作者：DukeWF  \* @ 创建时间：2018-04-30  \* @ 版本编号：1.00  \*\*/  **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: getDataSet  \* @ 功能描述：根据查询SQL语句进行查询操作。  \* @ 传入参数：用于查询的SQL语句sql  \* @ 返回类型：(ArrayList<HashMap<String, String>>)  \* @ 文件作者：DukeWF  \* @ 创建时间：2018-04-30  \* @ 版本编号：1.00  \*\*/  **public** **static** ArrayList<HashMap<String, String>> getDataSet(String sql) {  HashMap<String, String> hash = **null**;  ArrayList<HashMap<String, String>> list = **new** ArrayList<>();  ResultSetMetaData rsma = **null**;  **int** columncount = 0;  **try** {  conn = DBUtil.getConn();  st = conn.createStatement(ResultSet.TYPE\_SCROLL\_INSENSITIVE, ResultSet.CONCUR\_READ\_ONLY);  rs = st.executeQuery(sql);  rsma = rs.getMetaData();  **while** (rs.next()) {  hash = **new** HashMap<>();  columncount = rsma.getColumnCount();  **for** (**int** i = 1; i <= columncount; i++) {  hash.put(rsma.getColumnName(i), rs.getString(i));  }  list.add(hash);  }  } **catch** (SQLException e) {  e.printStackTrace();  } **finally** {  finallyHandle(conn, st, rs);  }  **return** list;  }  /\*\*  \* @ 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>>)  \* @ 文件作者：DukeWF  \* @ 创建时间：2018-05-06  \* @ 版本编号：1.00  \*/  **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：finallyHandle  \* @ 功能描述：对数据库操作结束进行资源释放工作。  \* @ 传入参数：当前连接conn、状态st、结果集rs  \* @ 返回类型：void  \* @ 文件作者：DukeWF  \* @ 创建时间：2018-04-30  \* @ 版本编号：1.00  \*\*/  **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();  }  }  演示：    图2 数据库测试结果  3、实现系统的登录功能  前端代码：  **登录页**  <%@ page language=*"java"* import=*"java.util.\*"* pageEncoding=*"UTF-8"*%>  <!DOCTYPE html>  <html>  <head>  <title>登录</title>  <link rel=*"stylesheet"*  href=*"https://maxcdn.bootstrapcdn.com/font-awesome/4.5.0/css/font-awesome.min.css"*>  <link rel=*"stylesheet"*  href=*"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"*>  <link rel=*"stylesheet"* type=*"text/css"*  href=*"*${pageContext.request.contextPath}*/css/Lab2\_login.css"*>  </head>  <body>  <div class=*"container"*>  <div class=*"row"*>  <div class=*"col-md-offset-3 col-md-6"*>  <form class=*"form-horizontal"*  action=*"*${pageContext.request.contextPath}*/LoginServlet"*  method=*"post"*>  <span class=*"heading"*>用户登录</span>  <div class=*"form-group"*>  <input type=*"text"* class=*"form-control"* id=*"username"* name=*"username"*  placeholder=*"用户名"*> <i class=*"fa fa-user"*></i>  </div>  <div class=*"form-group help"*>  <input type=*"password"* class=*"form-control"* id=*"password"* name=*"password"*  placeholder=*"密　码"*> <i class=*"fa fa-lock"*></i> <a href=*"#"*  class=*"fa fa-question-circle"*></a>  </div>  <div class=*"form-group"*>  <div class=*"main-checkbox"*>  <input type=*"checkbox"* value=*"None"* id=*"checkbox1"* name=*"check"* />  <label for=*"checkbox1"*></label>  </div>  <span class=*"text"*>Remember me</span>  <button type=*"submit"* class=*"btn btn-default"*>登录</button>    <button type=*"button"* class=*"btn btn-default"* onclick="location.href='reg.jsp'">注册</button>  </div>  </form>  </div>  </div>  </div>  <p>${msg}</p>  </body>  **登录状态页**  </html>  <%@ page language=*"java"* import=*"java.util.\*"* pageEncoding=*"UTF-8"*%>  <%  String path = request.getContextPath();  String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";  %>  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  <html>  <head>  <base href=*"*<%=basePath%>*"*>    <title>登录状态页</title>    </head>    <body>  <%  **if**(session.getAttribute("loginState")=="1"){  response.sendRedirect(".\\index.jsp");  }  **else**{  response.sendRedirect(".\\login.jsp");  }  %>  </body>  </html>  后端实现：  **protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  // **TODO** Auto-generated method stub  //doGet(request, response);  response.setContentType("text/html;charset=utf-8");  String username = WebUtil.getParameter(request,"username");  String password = WebUtil.getParameter(request,"password");  PrintWriter out = response.getWriter();    //System.out.println(username+","+password);    HashMap<String, String> hashMap = **new** HashMap<String, String>();  hashMap.put("username", username);  hashMap.put("password", password);    **if**(DBUtil.getDataCount("user",hashMap)==1) {  HttpSession session = request.getSession();  session.setAttribute("username",username); //用户名  session.setAttribute("loginState","1"); //登录状态  response.sendRedirect("./loginStateHandle.jsp");  }  **else** {  HttpSession session = request.getSession();  session.setAttribute("loginState","0"); //登录状态  response.sendRedirect("./loginCheck.jsp");  }  }  演示    图3-1 用户登录    图3-2 新用户注册    图3-3 用户未登录    图3-4 用户退出登录    图3-5 用户登录成功提示  4、主界面的设计及实现  前端实现：  <%@ page language=*"java"* import=*"java.util.\*"* pageEncoding=*"UTF-8"*%>  <%  String path = request.getContextPath();  String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";  %>  <!DOCTYPE html>  <html xmlns=*"http://www.w3.org/1999/xhtml"*>  <head>  <meta charset=*"utf-8"* />  <meta name=*"viewport"* content=*"width=device-width, initial-scale=1.0"* />  <title>main</title>  <!-- BOOTSTRAP STYLES-->  <link href=*"assets/css/bootstrap.css"* rel=*"stylesheet"* />  <!-- FONTAWESOME STYLES-->  <link href=*"assets/css/font-awesome.css"* rel=*"stylesheet"* />  <!--CUSTOM BASIC STYLES-->  <link href=*"assets/css/basic.css"* rel=*"stylesheet"* />  <!--CUSTOM MAIN STYLES-->  <link href=*"assets/css/custom.css"* rel=*"stylesheet"* />  <!-- GOOGLE FONTS-->  <link href=*'http://fonts.googleapis.com/css?family=Open+Sans'* rel=*'stylesheet'* type=*'text/css'* />  </head>  <body>  <div id=*"page-inner"*>  <div class=*"row"*>  <div class=*"col-md-12"*>  <h1 class=*"page-head-line"*>欢迎使用</h1>  <h1 class=*"page-subhead-line"* style="font-style: *normal*">在使用过程中，如有疑问，请联系管理员。</h1>  </div>  </div>  <!--/.ROW-->  <div class=*"row"*>  <div class=*"col-md-12"*>  <div class=*"panel panel-default"*>  <div class=*"panel-heading"*>  功能选项  </div>  <div class=*"panel-body"*>  <div class=*"row"*>  <div class=*"col-md-3 "* onclick="location='welcome.jsp'">  <div class=*"alert alert-info text-center"*>  <i class=*"fa fa-desktop fa-5x"*></i>  <h3>实验1</i>  </h3>  实现HTML静态界面与数据库连接操作<br>  </div>  </div>  <div class=*"col-md-3 "* onclick="location='Form.jsp'">  <div class=*"alert alert-success text-center"*>  <i class=*"fa fa-bars fa-5x"*></i>  <h3>实验2</h3>  实现登录、表单数据的提交，并使用JavaBean对数据进行封装  </div>  </div>  <div class=*"col-md-3 "* onclick="location='DataGrid.jsp'">  <div class=*"alert alert-warning text-center"*>  <i class=*"fa fa-fax fa-5x"*></i>  <h3>实验3</h3>  <br> 实现easyUI框架DataGrid的使用  </div>  </div>  <div class=*"col-md-3 "* onclick="location='welcome.jsp'">  <div class=*"alert alert-danger text-center"*>  <i class=*"fa fa-bomb fa-5x"*></i>  <h3>实验4</h3>  <br> 实现SSMS管理系统  </div>  </div>  </div>  </div>  </div>  </div>  </div>  <!--/.ROW-->  </div>  </body>  演示：    图4-1 用户信息提示模块    图4-2 主界面（1920\*1080分辨率下）    图4-3 主界面（分屏模式下）  5、具体模块的功能实现  增加信息：  后端：  **case** "add":  Student student\_insert = **new** Student();  student\_insert.setAddress(request.getParameter("address"));  student\_insert.setAge(request.getParameter("age"));  student\_insert.setBirthday(request.getParameter("birthday"));  student\_insert.setDemo(request.getParameter("demo"));  student\_insert.setInstitute(request.getParameter("institute"));  student\_insert.setPhone(request.getParameter("phone"));  student\_insert.setSname(request.getParameter("sname"));  student\_insert.setSno(Integer.*parseInt*(request.getParameter("sno")));  student\_insert.setPoliticalstate(request.getParameter("politicalstate"));    InsertStudentInfo(response, student\_insert);  **break**;  **private** **void** InsertStudentInfo(HttpServletResponse response, Student student) {  **try** {  JSONObject jsonObject = **new** JSONObject();  ArrayList<String> List = **new** ArrayList<>();  String insert = "INSERT INTO student(sno,sname,age,politicalstate,birthday,address,phone,institute,demo) VALUES('";  insert+=student.getSno()+"','"+student.getSname()+"','"+student.getAge()+"','"+student.getPoliticalstate()+"','"+student.getBirthday()+"','"+student.getAddress()+"','"+student.getPhone()+"','"+student.getInstitute()+"','"+student.getDemo()+"')";  System.out.println(insert);  **boolean** result = DBUtil.executeBatch(insert);  PrintWriter out = response.getWriter();  response.setCharacterEncoding("utf-8");  **if**(result){  jsonObject.put("ret", "1");  System.out.println("插入成功");  }  **else** {  jsonObject.put("ret", "0");  jsonObject.put("reason", "数据库操作失败");  System.out.println("插入失败");  }  out.print(jsonObject);  out.close();  } **catch** (Exception ex) {  ex.printStackTrace();}  }  演示：    图5-1-1 添加学生信息    图5-1-2 添加成功提示  修改信息  后端：  **case** "edit":  **int** oldsno = Integer.*parseInt*(request.getParameter("oldsno"));  Student student\_update = **new** Student();  student\_update.setAddress(request.getParameter("address"));  student\_update.setAge(request.getParameter("age"));  student\_update.setBirthday(request.getParameter("birthday"));  student\_update.setDemo(request.getParameter("demo"));  student\_update.setInstitute(request.getParameter("institute"));  student\_update.setPhone(request.getParameter("phone"));  student\_update.setSname(request.getParameter("sname"));  student\_update.setPoliticalstate(request.getParameter("politicalstate"));    UpdateStudentInfo(response, student\_update, oldsno);  **break**;  **private** **void** UpdateStudentInfo(HttpServletResponse response, Student student, **int** oldsno) {  **try** {  JSONObject jsonObject = **new** JSONObject();  ArrayList<String> List = **new** ArrayList<>();    String update = "UPDATE student SET ";  update+="sname = '"+student.getSname()+"',";  update+="birthday = '"+student.getBirthday()+"',";  update+="age = '"+student.getAge()+"',";  update+="politicalstate = '"+student.getPoliticalstate()+"',";  update+="address = '"+student.getAddress()+"',";  update+="phone = '"+student.getPhone()+"',";  update+="institute = '"+student.getInstitute()+"',";  update+="demo = '"+student.getDemo()+"'";  update+=" WHERE sno='"+oldsno+"'";  **boolean** result = DBUtil.executeBatch(update);  PrintWriter out = response.getWriter();  response.setCharacterEncoding("utf-8");  **if**(result){  jsonObject.put("ret", "1");  System.out.println("更新成功");  }  **else** {  jsonObject.put("ret", "0");  jsonObject.put("reason", "数据库操作失败");  System.out.println("更新失败");  }  out.print(jsonObject);  out.close();  } **catch** (Exception ex) {  ex.printStackTrace();  }  }  演示：    图5-2-1 修改学生信息    图5-2-2 修改成功提示  删除信息：  后端代码：  **private** **void** deleteStudentInfo(HttpServletResponse response, String sno) {  **try** {  JSONObject jsonObject = **new** JSONObject();  ArrayList<String> List = **new** ArrayList<>();  String delete = "DELETE FROM student WHERE sno='" + sno + "'";  List.add(delete);  **boolean** result = DBUtil.executeBatch(List);  PrintWriter out = response.getWriter();  response.setCharacterEncoding("utf-8");  **if**(result){  jsonObject.put("ret", "1");  System.out.println("删除成功");  }  **else** {  jsonObject.put("ret", "0");  jsonObject.put("reason", "数据库操作失败");  System.out.println("删除失败");  }  out.print(jsonObject);  out.close();  } **catch** (Exception ex) {    ex.printStackTrace();  }  }  演示：    图5-3 删除成功提示  查找特定信息：  后端代码：  **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;  }  演示：    图5-4 查询选项  心得体会：  通过本次实验，我初步了解了利用扩展的JavaScript框架构架Web应用程序的方法，并学习掌握了EasyUI中常用组件的使用方法，包括下拉列表（ComboBox）与表格（DataGrid）控件。此外我也掌握jQuery常用方法的使用，并对JSON格式串和Java集合工具类的转换工具的使用有了一定的了解。Web开发包括前后端与数据库等方面，通过前三个实验的学习，我能够初步实现一个有较好交互界面的Web应用程序。虽然还有很多问题亟需解决，但在学习的过程中我的编程能力有了较好的提升。这对于日后的编程课程与其他专业课程的学习，乃至日后的工作与科研，都有着较为关键的帮助。 | | | | | | | | |