## 通过案例快速回顾Java Web核心技术

海量数据最终需要通过Web技术进行展示，因此本章先通过一个“分页”案例对数据库及Java Web的核心知识进行回顾。

### 1 详解分页原理与分页SQL

试想，如果一个页面中有几千、几万、几十万、甚至几百万条数据，那么页面一定会被拉的很长，用户必须拉滚动条才能访问全部数据。显然，将如此多的数据放在同一个页面中，会降低系统的查询性能，并增大页面的负荷。因此，就需要通过分页技术来控制数据量的查询：例如，可以要求每一页仅显示10条数据，并且第一次也只从数据库中查询10条数据；当用户点击“上一页”或“下一页”时，再查询新页面的10条数据。也就是说，在每次翻页的时候，只从数据库里查询出当前页所需要的数据。可以发现，分页的实现依赖于具体的SQL语句，但不同数据库的SQL语句之间存在着差异。因此在编写分页时，我们需要对不同的数据库，写不同的SQL语句，本章采用的是Oracle数据库，以下是分页的具体实现。

要想实现分页，就需要提供以下5个属性（变量）。

**（1）数据总数：**一共有多少条数据需要显示；可以通过”select count(1) from …”从数据库表中获取。

**（2）页面大小：**每页显示几条数据；可以由用户自己设置。

**（3）总页数：**总页数可以由“数据总数”和“页面大小”计算得出。例如，①如果一共有80条数据（即“数据总数”为80），而每页只显示10条（即“页面大小”为10），则可以得到总页数=80/10=8（正好除尽，没有余数），共8页；②如果一共有82条数据，每页仍然显示10条，则总页数=82/10（有余数）+1=9，共9页数据。因此，我们在求总页数之前，需要先判断是否有余数。综上，求总页数的公式：

总页数 = （数据总数 % 页面大小==0 ） ? （数据总数 / 页面大小） : （数据总数 / 页面大小 +1）

需要注意，因为“总页数”是由“数据总数”和“页面大小”计算而来的，所以不应该手动的为“总页数”赋值，即不存在“总页数”的setter方法。

**（4）当前页的页码：**指定需要显示第几页的数据**；**可以由用户自己指定。

**（5）实体类集合：**如List<Student> students，用来保存当前页面中全部学生的信息。

为了便于维护，我们可以把这5个属性封装到一个Page工具类中，并提供相应的getter/setter方法，如下所示。

【源码：demo/ch14/org/student/entity/Page.java】

com.yanqun.student.entity.Page.java

|  |
| --- |
| // 省略import  public class Page  {  // 总页数  private int totalPage;  // 数据总数;即一共有都少条数据，需要显示  private int totalCount;  // 页面大小;即每页显示几条数据  private int pageSize;  // 当前页的页码  private int currentPage;  // 实体类集合;如List<Student> students，用来保存当前页面中全部学生的信息  private List<Student>students;  /\* 不存在“总页数”的setter方法，因为总页数是由“数据总数”和“页面大小”计算而来的  当“页面大小”和“数据总数”被赋值之后，可以自动计算出“总页数”  \*/  public void setTotalCount(inttotalCount)  {  this.totalCount = totalCount;  //自动计算出“总页数”  totalPage = this.totalCount % pageSize == 0 ?  (this.totalCount / pageSize) : this.totalCount / pageSize + 1;  }  //省略其他常规的setter/getter方法  } |

再在数据访问层，增加分页操作需要的两个方法：

（1）获取数据总数的方法.

（2）获取当前页面中全部学生的信息的集合的方法（例如获取List<Student> students集合的值）。而这两个方法都需要使用特定的SQL语句：

（1）获取数据总数方法：使用“select count(\*) from student”语句即可。

（2）获取当前页面中全部学生的信息的集合：

我们需要知道当前页的第一条及最后一条数据的行号，然后使用“select \* from student where 编号>=第一条数据行号 and 编号<=最后一条数据行号”即可查出当前页的全部学生信息。为了分析“第一条及最后一条数据的行号”，笔者绘制了如下的表14-1（假设每页显示10条数据，即页面大小为10）：

表14-1 每页的起止数据

|  |  |  |
| --- | --- | --- |
| 当前页 | 数据起止 | 推导公式（“数据起止”的等价写法） |
| 第1页 | 第1 ~ 10条 | 第(1-1)\*10+1 ~ 1\*10条 |
| 第2页 | 第11 ~ 20条 | 第(2-1)\*10+1 ~ 2\*10条 |
| 第3页 | 第21 ~ 30条 | 第(3-1)\*10+1 ~ 3\*10条 |
| 第n页 | - | 第(n-1)\*10+1 ~ n\*10条 |

可以发现，第n页需要显示的数据，就是第 (n-1)\*10+1条至第 n\*10条之间的数据，而其中的“10”就是“页面大小”。因此，第n页需要显示的数据范围如下：

第“ (n-1)\*页面大小+1”条 至 第 “n\*页面大小”条

所以，查询当前页的全部学生信息的SQL为：

|  |
| --- |
| select \* from  (  select rownum r,t.\* from  (select s.\* from student s orderby sno asc ) t  )  wherer<= 当前页的页码\*页面大小 and r>=((当前页的页码-1)\*页面大小+1) |

将上述SQL进行优化调整，可以写成如下的等价形式。

|  |
| --- |
| select \* from  (  select rownum r,t.\* from  (select s.\* from student s orderby sno asc ) t  where rownum<= 当前页的页码\*页面大小  )  where r>=((当前页的页码-1)\*页面大小+1) |

以上就是oracle的分页SQL语句。

### 2 使用三层架构实现基于三种数据库的分页案例

不难发现，在oracle的分页SQL语句中需要使用“当前页的页码（currentPage）”和“页面大小（pageSize）”两个参数，而这两个参数需要通过三层逐步传递：用户通过JSP输入或指定currentPage 和pageSize🡪在JSP中，将二者附加在超链接或表单中，传入表示层后端代码Servlet中🡪在Servlet中，将二者传入SERVICE层方法的入参中🡪再在SERVICE层中，将二者传入DAO层方法的入参中🡪最后在DAO层中，将二者放入分页的SQL语句之中，并通过DBUtil执行最终的SQL语句，从而实现分页。具体如下所述。（代码实现的顺序：数据库帮助类DBUtil🡪DAO层🡪SERVICE层🡪UI层）。

为了实现分页，DBUtil需要在数据库中查询数据总数和当前页的数据集合，如下所示。

（1）数据总数：即Page类中的属性totalCount。

（2）当前页面中学生信息的集合，即Page类中的属性students。

因此需要在DBUtil中加入“查询数据总数”方法getTotalCount()和“查询学生信息集合）”的方法executeQuery()，代码结构如下所示。

【源码：demo/ch14/org/student/util/DBUtil.java】

com.yanqun.student.util.DBUtil.java

|  |
| --- |
| ...  public class DBUtil  {  …  // 查询数据总数  public static int getTotalCount(String sql)  { ...}  // 查询当前页面中学生信息的集合  public static List<Student> executeQuery(String sql,Object[] params)  { ...}  } |

再在DAO层，加入“查询数据总数”方法，和“查询当前页面中全部学生信息的集合”的方法，如下所示。

（1）DAO接口

通过getTotalCount()“查询数据总数”，并通过getStudentsListForCurrentPage()“查询当前页面中全部学生信息的集合”，如下所示。

【源码：demo/ch14/org/student/dao/IStudentDao.java】

com.yanqun.student.dao.IStudentDao.java接口

|  |
| --- |
| public interface IStudentDao  {  // 获取“数据总数”  public int getTotalCount();  // 获取“当前页面中全部学生信息的集合”，用来给Page中的集合属性students赋值;currentPage表示当前页的页码，pageSize表示页面大小  public List<Student> getStudentsListForCurrentPage(int currentPage  , int pageSize);  …  } |

（2）DAO实现类

编写StudentDaoImpl，用于实现IStudentDao中定义的方法，如下所示。

【源码：demo/ch14/org/student/dao/impl/StudentDaoImpl.java】

com.yanqun.student.dao.impl.StudentDaoImpl.java实现类

|  |
| --- |
| public class StudentDaoImpl implements IStudentDao  {  …  public int getTotalCount()  {  String sql = "select count(\*) from student ";  return DBUtil.getTotalCount(sql);  }    /\*  获取第currentPage页的全部学生信息（每页显示pageSize条数据）  通过执行分页SQL语句实现  \*/  public List<Student> getStudentsListForCurrentPage(int currentPage, int pageSize)  {  String sql = "select \* from " + "(" + "select rownum r,t.\* "  + "from (select s.\* from student s order by stuno asc ) t "  + "where rownum<= ? )" + "where r>= ?";  Object[] os =  { currentPage \* pageSize, (currentPage - 1) \* pageSize + 1 };    // 获取当前页的学生集合  ResultSet rs = DBUtil.executeQuery(sql, os);  List<Student>students = new ArrayList<Student>();  try  {  while (rs.next())  {  int sNo = rs.getInt("stuNo");  String sName = rs.getString("stuName");  int sAge = rs.getInt("stuAge");  String gName = rs.getString("graName");  // 将查到的学生信息，封装到stu对象中  Student stu = new Student(sNo, sName, sAge, gName);  // 将封装好的stu对象，存放到List集合中  students.add(stu);  }  }  ...  return students;  }  } |

然后在SERVICE层，加入“获取数据总数”方法，和“获取当前页面中全部学生信息的集合”方法，如下所示。

（1）SERVICE接口

编写业务逻辑层接口，用于定义分页查询的相关方法，如下所示。

【源码：demo/ch14/org/student/service/IStudentService.java】

com.yanqun.student.service.IStudentService.java接口

|  |
| --- |
| public interface IStudentService  {  // 获取“数据总数”  public int getTotalCount();  //“获取当前页面中学生信息的集合”  public List<Student> getStudentsListForCurrentPage(int currentPage, int pageSize);  …  } |

（2）SERVICE实现类

编写业务逻辑层的实现类中，在其中调用数据访问层已实现的查询方法，如下所示。

【源码：demo/ch14/org/student/service/impl/StudentServiceImpljava】

com.yanqun.student.service/impl/StudentServiceImpl.java

|  |
| --- |
| public class StudentServiceImpl implements IStudentService  {  // 获取“数据总数”  public int getTotalCount()  {  return stuDao.getTotalCount();  }  public List<Student> getStudentsListForCurrentPage(int currentPage, int pageSize)  {  return stuDao.getStudentsListForCurrentPage(currentPage, pageSize);  }  …  } |

不难发现，SERVICE层和DAO层，都需要currentPage(当前页)和pageSize（页面大小）两个参数。现在，我们就通过表示层来获取这两个参数。

（1）表示层的后台代码

先在表示层的后台代码（Servlet）中加入控制页码的程序，然后给Page类的各个属性赋值，最后再跳转到表示层的前台JSP中，如下所示。

【源码：demo/ch14/org/student/servlet/QueryAllStudentsServlet.java】

com.yanqun.student.servlet.QueryAllStudentsServlet.java

|  |
| --- |
| package org.yanqun.servlet;  //省略import  public class QueryAllStudentsServlet extends HttpServlet  {  …  protected void doPost(HttpServletRequest request  , HttpServletResponse response)  throws ServletException, IOException  {  request.setCharacterEncoding("UTF-8");  // 获取前台传来的当前页码，即currentPage值  String curPage = request.getParameter("currentPage");  // 如果curPage值为null，说明是第一次进入此Servlet，  则将curPage设为第1页  if (curPage == null)  {  curPage = "1";  }  int currentPageNo = Integer.parseInt(curPage);  // 调用业务逻辑层代码  IStudentService stuService = new StudentServiceImpl();  // 获得总记录数  int totalCount = stuService.getTotalCount();  // 获取分页帮助类  Page pages = new Page();  // 设置页面大小，即每页显示的条数（本次，假设每页显示3条数据）  pages.setPageSize(3);  // 设置总记录数  pages.setTotalCount(totalCount);  // 获取总页数  int totalpages = pages.getTotalPage();  // 对首页与末页进行控制：页数不能小于1，也不能大于最后一页的页数  if (currentPageNo< 1)  {  currentPageNo = 1;  }  else if (currentPageNo>pages.getTotalPage())  {  currentPageNo = totalpages;  }  // 设置当前页的页码  pages.setCurrentPage(currentPageNo);  //调用业务逻辑层的方法，来获取当前页面中全部学生信息的集合  List<Student>students =  stuService.getStudentsListForCurrentPage(pages.getCurrentPage()  , pages.getPageSize());  // 设置每页显示的集合  pages.setStudents(students);  // 将存放当前页全部数据的对象pages，放入request作用域中。  即采用分页后，数据是通过分页帮助类Page的对象来传递的。  request.setAttribute("pages", pages);  // 跳转到首页（学生列表页）  request.getRequestDispatcher("index.jsp")  .forward(request, response);  }  } |

阅读以上代码可知， currentPage(当前页)的值，是通过前台JSP传来的 currentPage设置的；而pageSize（页面大小）的值，是通过硬编码的方式直接写成了3（读者也可以尝试将pageSize的值通过前台传来）。

因为程序是先执行QueryAllStudentsServlet，然后再跳转到显示页面index.jsp中，因此需要将QueryAllStudentsServlet设置为项目的默认启动程序，如下：

【源码：demo/ch14/WebContent/WEB-INF/web.xml】

WebContent/WEB-INF/web.xml

|  |
| --- |
| <?xml…>  <welcome-file-list>  <welcome-file>QueryAllStudentsServlet</welcome-file>  </welcome-file-list>  …  <servlet>  <servlet-name>QueryAllStudentsServlet</servlet-name>  <servlet-class>  org.yanqun.servlet.QueryAllStudentsServlet  </servlet-class>  </servlet>  <servlet-mapping>  <servlet-name>QueryAllStudentsServlet</servlet-name>  <url-pattern>/QueryAllStudentsServlet</url-pattern>  </servlet-mapping>  </web-app> |

（2）表示层的前台代码

最后，再在表示层的前台代码中，获取Servlet传来的Page类对象pages，通过pages获取到当前页的学生数据集合等信息，最后再通过用户的点击，来设置currentPage的值.

【源码：demo/ch14/WebContent/index.jsp】

WebContent/index.jsp

|  |
| --- |
| <body>  …  <table border="1">  <tr>  <th>学号</th>  <th>姓名</th>  <th>年龄</th>  <th>操作</th>  </tr>    <%  //获取带数据的分页帮助类对象  Page pages=(Page)request.getAttribute("pages");  //总页数  int totalpages=pages.getTotalPage();  //当前页的页码  int pageIndex=pages.getCurrentPage();    //获取当前页中的学生数据集合  List<Student> students =pages.getStudents();  if(students != null)  {    for(Student stu : students)  {  %>  <tr>  <td>  <ahref="QueryStudentByNoServlet?stuNo  =<%=stu.getStudentNo() %>">  <%=stu.getStudentNo() %>  </a>  </td>  <td><%=stu.getStudentName() %></td>  <td><%=stu.getStudentAge() %></td>    <td><ahref="DeleteStudentServlet?stuNo  =<%=stu.getStudentNo() %>">删除  </a>  </td>  </tr>  <%  }  }  %>  </table>  当前页数：[<%=pageIndex%>/<%=totalpages%>]  <%  //只要不是首页，则都可以点击“首页”和“上一页”  if(pageIndex > 1)  {  %>  <%-- 通过用户点击超链接，将页码传递给Servlet --%>  <a href="QueryAllStudentsServlet?currentPage=1">首页  </a>&nbsp;  <a href="QueryAllStudentsServlet?currentPage  =<%= pageIndex -1%>">上一页  </a>  <% }  //只要不是末页，则都可以点击“下一步”和“末页”  if(pageIndex < totalpages)  {  %>  <%-- 通过用户点击超链接，将页码传递给Servlet --%>  <a href="QueryAllStudentsServlet?currentPage  =<%= pageIndex +1%>">下一页  </a>  <a href="QueryAllStudentsServlet?currentPage  =<%=totalpages%>">末页  </a>  <%  }  %>  <a href="addStudent.jsp">增加</a>  </body> |

至此，就可以将分页SQL需要的两个参数“当前页的页码”及“页面大小”全部得到，也就完整的实现了分页功能。

本程序的运行结果如图14-1所示。

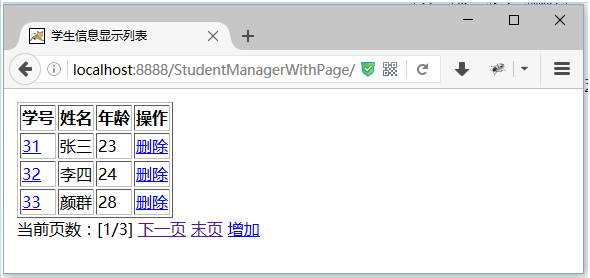


图14-1 分页查询结果

说明：

对于不同数据库的分页操作，唯一不同的就是SQL语句。除了oracle以外，其他常用数据库（如MySql、SqlServer）中的分页SQL语句如下所示。

（1）MySql:

select \* from 表名 limit (当前页的页码-1)\*页面大小,页面大小;

1. SqlServer

①方式一：

select top 页面大小 \* from 表名

where id not in(select top (当前页的页码-1)\*页面大小 id from 表名)

其中id表示“数据表的唯一标示符”。

②方式二（SqlServer2005后支持）：

select \*from

(

select row\_number() over (sno order by 编号 asc) as r,\* from 表名

where r<=当前页的页码\*页面大小

)

where r>=(当前页的页码-1)\*页面大小+1 ;

此种分页sql与oralce的分页sql非常相似，二者的区别：

a oracle中使用的伪劣是rownum ，SqlServer使用的是row\_number() ；

b oracle需要排序（为了排序，单独写了一个子查询），但是在SqlSserver 中可以省略该排序的子查询，因为SqlServer可以通过over直接排序。

③方式三（SqlServer2012后支持）：

select \* from 表名oreder by 编号

offset (当前页的页码-1)\*页面大小+1 rows fetch next 页面大小 rows only ;