数据库高级开发与专业应用系列

Java

数据库开发与专业应用

敬 铮 主编　　李 伟 编著

国防工业出版社

http://www.ndip.com.cn

数据库高级开发与专业应用系列

Java 数据库开发与专业应用

敬铮　主编

李伟　编著

国防工业出版社

·北京·

前　言

随着 Internet/Intranet 的普及以及 WEB 技术的发展，人们对信息的需求越来越强烈，数据库与 Internet/Intranet 应用软件的集成已经成为了非常迫切的问题。互联网上数据库产品的复杂，有限的带宽，以及互联网上软件产品的跨平台性，将是我们遇到的最大困难。

Java 凭借其强大的网络功能赢得了网络编程语言的美称，最为引人注目的可能要数它的与平台无关的特性了。仅凭这一点，足以另其他的传统的过程式语言汗颜。它提供了 C,C++,SmallTalk, BASIC, COBOL 和 4GLs 的许多优点。

SUN 公司给 Java 添加了许多更为强大的功能，其中最为抢眼的当数 JDBC——这一强大的数据库访问功能。

JDBC 提供了灵活的数据库访问机制，可以在客户端进行几乎所有与数据库相关的操作——建立/删除数据表、视图、索引等。JDBC 可以连接现有的 ODBC 数据库，它可以适应当今流行的各种数据库系统和平台。能够非常平滑地进行移植。JDBC API 既支持数据库访问的两层模型，同时也支持三层模型。

本书将会向读者全面介绍 JDBC 的相关知识和实际应用开发，同时还会介绍时下刚刚流行的网络脚本语言 JSP 和 Web 服务器端的 SERVERLET 程序的开发与应用。

本书主要内容如下：

第 1 章：简单介绍 Java 的历史、发展、前景等，希望能够使读者对于 Java 语言有一个整体上的把握。

第 2 章：介绍了 Java 的开发环境——JDK，虽然目前有很多开发工具和集成的开发环境，但是它们基本上是在 JDK 的基础上进行集成的，掌握最核心的东西才是我们的目的，所以，本书采用了最原始的开发工具——JDK。

第 3 章：简要介绍 Java 语言，Java 语言基本上同 C/C++的语法大体上类似，但是也有很多独特之处，本章对于 Java 语言进行了简单的介绍。

第 4 章：数据库的基础知识，作为一本介绍数据库开发的书籍，我们认为关于 DBMS 数据库的基础理论知识是必不可少的。

第 5 章：简要介绍 JDBC——Java 的数据库访问类库，使得读者能够从总体上把握 JDBC 的各种功能。

第 6 章：讲解如何使用 JDBC 进行数据库连接操作，包括 CONNECTION 等对象的用法介绍。

第 7 章：讲解使用 JDBC 进行数据库查询操作，包括 STATEMENT 等对象的介绍，本章是掌握 JDBC 的关键部分，也是 JDBC 的精华所在。

第 8 章：向读者介绍使用 JDBC 进行数据库定义操作，包括了建立数据库、数据表等使用的操作。这些涉及到系统管理员的相关知识。

第 9 章：本章介绍 Web 服务器端的程序开发以及相关的例子程序，本章对于 Servlet

进行了简单介绍。

第 10 章：介绍了如何使用 Servlet 进行交互式网络应用开发，包括了 HTML 的相关知识，同时给出了几个有代表性的例子程序。

第 11 章：向读者详细介绍了与 Web 应用相关的数据库开发。主要是讲 JDBC 同 SERVERLET 进行结合的方法，介绍附带有详尽的例子程序。

第 12 章：在前面介绍的基础上，我们这里面给出了几个实际中经常应用的例子程序，供读者提高水平之用。

第 13 章：介绍了刚刚流行的网络脚本语言 JSP。

第 14 章：最后我们给出了 JDBC 开发过程中经常使用的几个类库的参考。

本书主要面向广大网络应用程序的开发者以及对于 Java 的数据库编程感兴趣的读者，本书内容比较全面，包容了目前 Java 数据库的方方面面，既可以作为教材也可以作为一般的参考手册来使用，对于编程高手也有一定的参考价值。

全书由敬铮主编，李伟编著，邹忠望、刘朋、阎永军、刘敏、耿永兵、徐宏龙、李明、邓刚、王筑军、邢安大、胡小超、姚俊波、谢石、王小伟、颜林灿等同志参与了部分章节内容的编写和程序调试工作。由于时间仓促，作者水平有限，恳请广大专家读者不吝赐教。

需要全本请访问 www.ertongbook.com

目　录

第 1 章　Java 概述 …… 1
1.1　Java 的起源 …… 1
1.2　Java 的发展现状 …… 2
1.3　Java 的特点 …… 3
1.4　Java 的应用程序简介 …… 5
1.5　小结 …… 9
第 2 章　Java 开发环境 …… 10
2.1　JDK 简介 …… 10
2.2　JDK 的安装 …… 10
2.3　JDK 的配置 …… 13
2.4　小结 …… 13
第 3 章　Java 语言 …… 15
3.1　第一个简单的程序 …… 15
3.2　Java 的数据类型 …… 16
3.3　Java 的注释风格 …… 17
3.4　Java 的标识符 …… 18
3.5　Java 的分隔符 …… 18
3.6　Java 的运算符 …… 19
3.7　Java 的控制语句 …… 23
3.8　Java 的数组 …… 27
3.9　Java 的类 …… 31
3.10　Java 中的异常处理 …… 36
3.11　Java 的多线程 …… 38
3.12　小结 …… 42
第 4 章　数据库基础知识 …… 43
4.1　关系数据库概述 …… 43
4.1.1　关系数据库产生的历史 …… 43
4.1.2　关系数据库的定义 …… 43
4.1.3　关系数据库的基本术语 …… 44
4.1.4　关系数据库的特点 …… 45
4.2　SQL 语言 …… 46
4.2.1　SQL 的历史 …… 46
4.2.2　SQL 的简单查询功能 …… 46

4.3 SQL 的多表查询 ····· 58
4.4 子查询操作 ····· 62
4.5 SQL 的数据操作功能 ····· 67
4.6 创建存储过程 ····· 77
4.7 小结 ····· 81
第 5 章 JDBC 概述 ····· 82
5.1 JDBC 简介 ····· 82
5.2 Java 的数据库模型 ····· 83
5.3 同 SQL 语言的一致性 ····· 84
5.4 JDBC 与 ODBC 和其他 API 的比较 ····· 84
5.5 JDBC 的产品 ····· 85
5.6 JDBC 使用场合 ····· 86
5.7 JDBC 的安全性 ····· 87
5.8 JDBC 的开发环境 ····· 89
5.9 小结 ····· 89
第 6 章 使用 JDBC 进行数据连接 ····· 90
6.1 建立数据源 ····· 90
6.2 使用 JDBC 与数据库建立连接 ····· 95
6.3 小结 ····· 101
第 7 章 使用 JDBC 对数据库进行操作 ····· 102
7.1 数据插入 ····· 102
7.1.1 一个简单的插入例子 ····· 102
7.1.2 代码详解 ····· 103
7.1.3 关于 Statement ····· 104
7.2 数据查询 ····· 106
7.3 关于 ResultSet 对象 ····· 114
7.4 使用 JDBC 进行数据删除操作 ····· 118
7.5 使用 JDBC 进行数据更新操作 ····· 125
7.6 使用 PreparedStatement 对象 ····· 131
7.6.1 PreparedStatement 对象概述 ····· 131
7.6.2 一个例子程序 ····· 132
7.6.3 PreparedStatement 对象用法详解 ····· 138
7.7 使用 CallableStatement 对象 ····· 140
7.7.1 CallableStatement 概述 ····· 140
7.7.2 一个简单的例子 ····· 141
7.7.3 CallableStatement 对象的使用方法 ····· 142
7.8 小结 ····· 144
第 8 章 使用 JDBC 进行数据库定义操作 ····· 146
8.1 使用 JDBC 建立数据表 ····· 146
8.2 使用 JDBC 释放数据表 ····· 148

8.3 建立和删除视图……150
8.3.1 建立视图……150
8.3.2 视图的删除……153
8.4 小结……155
第 9 章 服务器端程序设计初步……156
9.1 Servlet 简介……156
9.2 Servlet 应用范围……156
9.3 Servlet 的结构……157
9.4 Servlet 的生命周期……157
9.5 Servlet 运行及开发环境的建立……158
9.6 Servlet 入门……160
9.7 Servlet 进阶……163
9.8 小结……166
第 10 章 开发 Web 交互式应用程序……167
10.1 网页元素简介……167
10.2 使用 Servlet 开发交互式网络应用程序……177
10.3 小结……184
第 11 章 开发 Web 数据库应用程序……186
11.1 在 Servlet 中进行数据库连接操作……186
11.2 使用 Servlet 进行数据库查询操作……188
11.3 在 Servlet 中插入数据……190
11.4 在 Servlet 中进行数据库删除……193
11.5 在 Servlet 中进行数据库更新操作……196
11.6 在 Servlet 中使用 PreparedStatement 对象……199
11.7 在 Servlet 中使用 CallableStatement 对象……203
11.8 小结……206
第 12 章 Servlet 综合实例……208
12.1 留言版实例……208
12.2 网站计数器实例……220
12.3 人才市场实例……223
12.4 小结……275
第 13 章 Java Server Pages……276
13.1 JSP 概述……276
13.2 JSP 语法讲解……277
13.2.1 语法概貌……277
13.2.2 JSP 语法详解……278
13.3 简单的 JSP 的例子……292
13.4 JSP 与 JDBC……330
13.5 小结……332
第 14 章 JDBC API 参考……333

14.1 Class DriverManager 333
14.2 Interface Driver 336
14.3 Class DriverPropertyInfo 337
14.4 Interface Connection 339
14.5 Interface PreparedStatement 344
14.6 Interface ResultSet 352

第 11 章　开发 Web 数据库应用程序

前面几章已经向大家讲解了 JDBC 和 Servlet 的基础知识,本章介绍这两种工具结合起来的方法。众所周知，目前网络上的应用后台一般都以数据库作为支撑，Servlet 也不例外，但是它自身有着得天独厚的优势，以其高效的执行方式再加上 JDBC 强大的数据库访问能力，必将逐步取代曾经辉煌的 CGI 程序，成为新一代网络应用的开发利器。所以掌握了 Servlet + JDBC 就等于掌握了自己的未来。

本章将通过多个小实例程序来向大家介绍通过 Servlet 访问数据库的方法。

11.1　在 Servlet 中进行数据库连接操作

下面先给出一个在 Servlet 中进行数据库连接的例子。

例子代码如下：

```
import java.io.*;
import java.servlet.*;
import javax.servlet.*;
import java.net.URL;
import java.sql.*;
public class LinkData extends HttpServerlet
{
Connection con;
public void init(ServerletConfig conf) throws ServerletException
{
    super.init();//调用父类的初始化函数
    String l="jdbc:odbc:TEST";
    try
    {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        con=DriverManager.getConnection(l,"nkwht","");
        }
         catch(Exception e)
         {
         //我们没有进行处理
        }
```

```
    }
    public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws IOException, ServletException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<body>");
        out.println("<head>");
        out.println("<title>这是一个使用 Serverlet 数据库连接的例子</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Serverlet 连接数据库</h1>");
        out.println("<pre>");
        try
        {
        DatabaseMetaData dma=con.getMetaData();
        out.println("连接的数据库:" + dma.getURL());
out.println("Driver     :"+ dma.getDriverName());
out.println("Version     :" + dma.getDriverVersion());
out.println("</pre>");
con.close();
}
catch(SQLException ex)
{
      while(ex!=null)
      {
      out.println("数据库异常被捕获了");
      out.println(ex.getSQLState());
      out.println(ex.getMessage());
     out.println(ex.getErrorCode());
      ex=ex.getNextException();
     }
}

        out.println("</body>");
        out.println("</html>");
    }
}
```

执行效果如图 11.1 所示。

需要全本请访问 www.ertongbook.com



图 11.1 数据连接的例子

代码详解：

这里的数据库连接的部分基本上同我们前面讲得差不多，不过需要注意的地方是我们进行驱动程序加载的代码已经放到了初始化函数 init 中，因为这个函数是在服务器第一次加载这个 Servlet 的时候使用的，以后不会再触发该函数，这样一来我们的程序的效率就可以大为提高。

代码的其余部分，可以参照我们前面讲结果的部分自行阅读，这里就不再赘述。

11.2 使用 Servlet 进行数据库查询操作

下面将给出几个关于数据库查询的例子使得大家能够对于在 Servlet 中进行相关的操作有一个比较清晰的认识。

代码如下：

```
import java.io.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.net.URL;
import java.sql.*;
public class SelectData extends HttpServlet
{
Connection con;
public void init(ServletConfig conf) throws ServletException
{
```

```
    super.init();//调用父类的初始化函数
    String l="jdbc:odbc:PUB";
    try
    {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        con=DriverManager.getConnection(l,"sa","");
        }
         catch(Exception e)
         {
         //没有进行处理
        }
    }
    public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws IOException, ServletException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<body>");
        out.println("<head>");
        out.println("<title>这是一个使用 Serverlet 数据库查询的例子</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Serverlet 查询数据库</h1>");
        out.println("<pre>");
        try
        {
        Statement stmt = con.createStatement();
ResultSet rs=stmt.executeQuery("Select * from TITLES");
while(rs.next())
{
    for(int i=1;i<11;i++)
    out.println("这是第" + i + " 个字段的内容 " + rs.getString(i));
}
con.close();
out.println("</pre>");
con.close();
}
catch(SQLException ex)
{
```

```
            while(ex!=null)
            {
            out.println("数据库异常被捕获了");
            out.println(ex.getSQLState());
            out.println(ex.getMessage());
            out.println(ex.getErrorCode());
            ex=ex.getNextException();
            }
        }

            out.println("</body>");
            out.println("</html>");
        }
    }
```

执行效果如图 11.2 所示。



图 11.2 查询数据库的例子

代码详解：

数据库驱动程序的加载同上面的例子，在 doGet 函数中进行查询，然后将查询的结果返回。标记<pre>含义是：表示数据输出保持格式不发生变化。

11.3 在 Servlet 中插入数据

这里给出一个进行数据插入的例子。

代码如下：

```
import java.io.*;
import java.text.*;
```

```
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.net.URL;
import java.sql.*;
public class InsertData extends HttpServlet
{
Connection con;
public void init(ServletConfig conf) throws ServletException
{
    super.init();//调用父类的初始化函数
    String l="jdbc:odbc:PUB";
    try
    {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        con=DriverManager.getConnection(l,"sa","");
        }
         catch(Exception e)
         {
         //没有进行处理
        }
    }
    public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws IOException, ServletException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<body>");
        out.println("<head>");
        out.println("<title>这是一个使用 Serverlet 数据库插入的例子</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Serverlet 插入数据库</h1>");
        out.println("<pre>");
        try
        {
        Statement stmt = con.createStatement();
        String sql="insert MyTable values('王洪涛',";
        for(int i=0;i<=100;i++)
```

```
            stmt.executeUpdate(sql + i +")");
ResultSet rs=stmt.executeQuery("Select * from MyTable");
while(rs.next())
{
      for(int i=1;i<3;i++)
      out.println("这是第" + i + " 个字段的内容 " + rs.getString(i));
}
con.close();
out.println("</pre>");
con.close();
}
catch(SQLException ex)
{
       while(ex!=null)
       {
       out.println("数据库异常被捕获了");
       out.println(ex.getSQLState());
       out.println(ex.getMessage());
      out.println(ex.getErrorCode());
       ex=ex.getNextException();
      }
}
          out.println("</body>");
          out.println("</html>");
     }
}
```

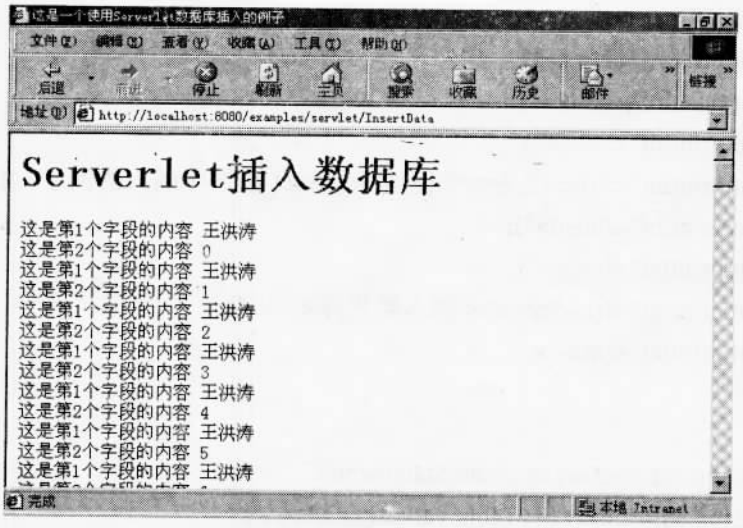


图 11.3 数据库插入的例子

代码详解：

这个例子创建的数据库中连续插入了 100 条记录，然后提取出来。使用了 Statement 对象的 executepdate 方法，利用了 ResultSet 对象。基本的思路同前面讲解过的例子基本一致。希望读者能够认真体会。

11.4　在 Servlet 中进行数据库删除

下面给出一个可以交互进行的数据库删除的例子。分为 HTML 页面和 Servlet 处理程序两部分。

HTML 代码如下：

```
<html>
<body>
<head>
<title>数据库删除的例子</title>
</head>
<body bgcolor="white">
<h1>数据库删除的例子</h1>
<form action="DeleteData" method=POST>
请你输入删除对象的分数:
<input type=text size=20 name=grade>
<br>
<input type=submit>
</form>
</body>
</html>
```

下面是 Servlet 的代码：

```
import java.io.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.net.URL;
import java.sql.*;
public class DeleteData extends HttpServlet
{
Connection con;
public void init(ServletConfig conf) throws ServletException
```

需要全本请访问 www.ertongbook.com