应用 JDBC 进行数据库编程

刘 江 (南京邮电学院 210003)

摘要:本文详细阐述了 Java 语言的 JDBC 数据库编程的方法。首先介绍了 JDBC 的几个常用的类,并对一些类方法做了说明,然后用一个这例来说明 JDBC 编程的大致步骤,最后简要分析了 JDBC 的未来发展情况。

关键词:Java JDBC Web DBMS SQL API CGI

毋庸置疑,数据库管理系统(DBMS)在当今各企业、政府部门以及其他一些社会团体中的重要性越来越明显了。而随着 Internet 的日益普及和 Intranet 的蓬勃发展,各企业纷纷建立 Web 页面,并使之与企业内部的数据库相连接,以提供对数据库更为方便快捷的访问。因此,如何有效地实现这一连接则成为很重要的一个问题。

一些数据库供应商开发出相应的工具软件来实现这一功能。其结构基本相同:Web 浏览器用户通过 HTML 表格输入静态或动态 SQL 查询;通过 API 或 CGI 调用,Web 服务器向 DBMS 代理提交表格;DBMS 代理将表格转化为数据库 SQL 查询;数据库查询结果转化为 HTML 格式,并通过 API 或 CGI 网关返回给 Web 服务器,再从 Web 服务器返回 Web 浏览器。

这些工具软件最大的缺点就是代码的专用性,不能适应多种数据库。

而 JavaSoft 开发的 JDBC 则消除了这一局限性。它提供了与平台无关的一套 API,实现了对数据库透明存取。假设一个程序员用 JDBC 编写一个职工工资的数据库应用程序,那么他不用考虑究竟要用哪个公司的数据库管理系统,如 Oracle、Sybase 或 Informix,因为你的程序将适用于任何支持具有 JDBC 驱动程序的数据库管理系统。

JDBC 即 Java DataBase Connectivity 的缩写。JavaSoft 开发它的目标主要是为 Java 定义一个 SQL 的接口,即所有对 JDBC 数据源文件的存取,均是通过 SQL 来实现。并且 JDBC 是实现于现有数据库接口之上的,从而使得 JDBC 用一个软件界面便可实现对现有关系数据库的访问,也就是所谓"透明访问"。这一特性也正体现了 Java 努力实现的代码可重用性。

接下来将就 JDBC 的实现机制做一详细分析。

一、JDBC 的组成

JDBC 分为两部分:JDBC API 和 JDBC 驱动 API。其中 JDBC 驱动 API 由驱动程序编写者或数据库供应商提供,应用程序开发者不用考虑。JDBC API 则是程序员最为有用的部分,它包括了许多的类和接口。最常用到的有五种:

DriverManager 类,负责管理各类数据库驱动;

Driver 类,定义了基本的数据库驱动程序的接口;

Connection 类,定义应用程序与数据库的连接过程;

Statement 类,在 Connection 完成后,用来执行单个 SQL 语句;

ResultSet 类,返回 Statement 的执行结果并对其进行处理。

下面分别介绍这几个类以及一些常用的接口函数。

二、类及其主要方法

1. DriverManager 类

如上所述,DriverManager 类提供管理一系列的 JDBC 驱动程序的服务。在初始化的时候它会调入所有在"jdbc.drivers"系统属性下指定的驱动器,然后跟踪可用的 JDBC 驱动器,保持对所有不同的数据库驱动程序的记录。在与数据库相连接时,它会自动搜寻适当的数据库驱动器。它通过保留一个 Vector 来存放所有驱动程序的有关信息。这里介绍几个常用的方法:

(1)public static synchronized Connection getConnection (String url, String user, String password) throws SQLException.,根据指定的用户名和口令,返回于指定 JDBC URL 的连接,该连接使用的是第一个能够处理该 URL 的驱动程序。

(2)public static synchronized Connection getConnection (String url) throws 的驱动程序。

(3) public static Driver getDriver (String url) throws SQLException, 从驱动程序注册表返回一个能够处理指定 JDBC url 的驱动程序。

(4) public static synchronized void registerDriver (Driver driver) throws SQLException, 把指定的驱动程序向 DriverManager 注册。

2. Driver 类

Driver 类提供了一般的数据库驱动程序和设置, 程序初始化期间装入, 在 Java 程序运行的期间驻留内存, 用于查询驱动器信息。当一个驱动程序调入后, 它应该生成一个自己的实例并在 DriverManager 中注册。具体的调用方法将在后面做详细的说明。

3. Connection 类

Connection 类将实际的数据库连接封装入一个易于使用的软件包中, 即提供简单的库函数来完成与某个数据库的连接。Connection 对象是指向数据库的指针, 相当于进入数据库的一个窗口。它包含有以下几个重要的方法:

(1) public abstract Statement createStatement() throws SQLException, 返回一个新的 Statement 对象。

(2) public abstract void close() throws SQLException, 立即释放连接所占用的 JDBC 和数据库资源。

(3) public abstract SQLWarnings getWarnings () throws SQLException, 返回包含第一条警告信息的 SQLWarnings 对象。

(4) public abstract boolean isClosed() throws SQLException, 若连接已关闭则返回真。

4. Statement 类

SQL 的查询、更新、删除等数据库操作的完成是通过 Statement 类来实现的。Statement 类代表了可在程序数据库间来回传递的数据库事项。它将 SQL 查询封装起来交给数据库, 不同的数据库通过相应的驱动程序将该查询进行转换, 以使之能够被处理, 然后返回结果。应用程序接受并以 ResultSet 类的形式保存该结果。它主要有以下几个方法:

(1) public abstract ResultSet executeQuery(String sql) throws SQLException, 执行指定语句并返回一个 ResultSet.

(2) public abstract int executeUpdate (String sql) throws SQLException, 执行 SQL INSERT, UPDATE, DELETE 语句, 返回行数。

(3) public abstract int getMaxRows () throws SQLException, 返回 ResultSet 的能够返回的最大行数, 若无限制则返回 0。

(4) public abstract void setMaxRows (int max) throws SQLException, 设置一个 ResultSet 中可以返回的最大行数(0 表示无限制)。

5. ResultSet 类

顾名思义, ResultSet 类是存放 Statement 的执行结果的数据结构, 同时也是对结果进行处理的类库。它封装了一个 SQL 的查询结果。这些结果是以数据行的形式存在的, 每一行包含有一个或多个列。对于数据行的操作是按顺序进行的, 而数据行中的各列则是可以随意地进行处理。该类中有许多对数据进行处理的方法, 一般具有 gettype()的形式, 这里的 type 为列中所需数据的类型, 返回来自结果集合的数据。如下面的一个方法:

public abstract String getString (int columnIndex) throws SQLException 它就是返回当前行指定的 columnIndex 列的值, 以字符串返回。

ResultSet 类中有一个很重要的方法, 在实际编程中经常用到:

public abstract boolean next () throws SQLException

它的作用是把下一行置为当前行, 读入新行时消除警告链, 无后续行返回假。因为 ResultSet 类保持着一个指针, 指向当前的数据行。初始状态下, 指针指向第一列之前。所以, 必须调用 next()方法移动指针指向第一个数据行。

JDBC 里还含有其他的类库, 这里就不一一阐述了。值得注意的是 ResultSetMetaData 类。

当需要对元数据的属性进行查询时会用到它。如方法

public abstract String getColumnLabel (int column) throws SQLException 返回列标题。

至此, 我们完成了对 JDBC 中的主要类的介绍, 下面将用实例来说明这些类的具体使用方法。

三、JDBC 编程实例

假定我们要对一个 ODBC 数据库文件进行操作, 文件名为 NBA50, 用户名为 Chicago, 通行字为 Bull。JDBC 用 URL 语法 jdbc:<subprotocol>:<subname>来识别 JDBC 连接, 那么我们的这个文件的 URL 地址应为:

jdbc:odbc:NBA50

请看下面的一段例程:

```
import java.applet.*;
```

```
import java.awt.*;
import java.io.*;
import java.sql.*;

public class testJdbc extends Applet{
  public void init(){
    try{
      //定义连接数据库的参数
      String myUrl = "jdbc:odbc:NBA50";
      String user = "Chicago";
      String password = "Bull";

      //连接数据库
      Connection myConnect = DriverManager.getConnection ( myUrl, user, password);
      Statement myOrder = myConnect.CreateStatement ();

      //对数据库进行操作
      ResultSet myFind = myOrder.executeQuery(
    " SELECT * FROM stars50 WHERE allscores>10000 ");
      while (myFind.next()){
        String name = myFind.getString( "starName" );
        int scores = myFind.getInt( "allscores" );
        System.out.println(name + " = = = " + Integer.toString(scores));
        }
      } catch (SQLException e){
        System.out.println( "SQL ERROR:" + e.toString());
        }
    myOrder.close();
    my.Connect.close();
  }}
```

我们来详细地分析一下这段程序。

程序中隐式地调入 Driver 类，通过 DriverManager.getConnection()完成。也可以显式地装载，用 Class 类的 forName ()方法指定载入某个特定的驱动程序，例如下列语句实现了装载一个名为"myDriver"的驱动程序到内存：

```
Class.forName("myDriver");
Driver usedDriver = new myDriver;
```

getConnection()同时还完成了连接指定的 URL 的工作。

接下来是生成 Statement 对象，对数据库操作它是必不可少的。然后是实际的查询过程，检索 NBA50 数据库中的 star50 表，返回总得分超过 10000 分的球员的数据，并打印在屏幕上。注意在这里使用了 next()函数，原因在前面以说明过了。

由于 JDBC 的各个类中的方法均可能抛出异常，所以必须用 try...catch...来处理可能出现的异常。程序中只是将异常信息输出。

最后是关闭 Statement 类和连接。

从这个程序中可以看出，用 JDBC 编程大致分四个步骤：

装载驱动程序 - - > 生成连接 - - > 查询数据库 - - > 关闭连接

四、JDBC 的缺点及未来发展

我们已经对 JDBC 编程有了一个大致的了解，也体会到了它的一些优点。而我认为 JDBC 最大的一个缺点就是由于采用了标准的编程接口，它必然会放弃某些数据库提供的一些独特和专门的功能，这可能导致应用程序执行性能低劣，甚至不可用。

但是，毕竟 JDBC 才问世不久，还处于逐渐完善的阶段，相信 Javasoft 公司必定会不遗余力地强化并推广它，因为没有它，Java 不可能成为一个平台型的开发语言，也不可能用于开发基于 Internet/Intranet 的 Client/Server 应用程序。而且 Java 的 applet 在 web 页面上表现极为出色，极大程度上扩展了 web 页面的交互性。Java 的这个特点正是开发一个良好的基于 Internet/Intranet 的 Client/Server 数据库应用程序所必需的。Java 的另外一个优势就在于能够实现跨平台运行，大大地减轻了开发人员的工作量。结合包括了 JDBC 在内的 Java Enterprise API，Java 语言的功能日益强大，相信一定会吸引越来越多的业界人士的关注与参与。

参考文献

[1] JDBC Guide, Sun Microsystems, Inc.

[2] Java API Documentation, Sun Microsystems, Inc.

[3] The Java Language Specification, Bill Joe and Guy Steele, Sun Microsystems, Inc.

[4] 用 Java 开发 Intranet 应用，Jerry Ablan，机械工业出版社。

（来稿时间：1998 年 5 月）