# 1.JDBC概念

JDBC技术规定了四种Java数据库操作的形式

1. JDBC-ODBC桥接技术(100%不用)，在Windows中有ODBC技术,指的是开放数据库连接,由微软提供的数据库连接应用,而JAVA简介利用JDBC技术间接操作ODBC技术,从而实现数据库的连接,流程:程序 -> JDBC -> ODBC ->数据库,性能不好,支持的版本最新的
2. JDBC直接连接:有不同的数据库生产商提供指定的数据库连接驱动程序,由于是JDBC直接操作数据库,所以是性能是最好的,但是支持的JDBC版本不是最新的
3. JDBC网络连接:使用专门的数据库的网络连接指定进行指定主机的数据库操作,此种方式最多
4. 模拟指定数据库的通讯协议自己编写数据库操作

Java连接任何数据库性能都是很高的,但是只有一个数据库性能是最差的,是SQLserver

# 2.数据库操作了介绍

所有类简介:

一个类:DriverManager类

四个接口:Connection Statement ResultSet PreparedStatement

操作流程:

1. 加载数据库的驱动程序(向容器加载)
2. 进行数据库连接(通过DriverManager类完成,Connection表示连接)
3. 进行数据的CRUD(Statement,PreparedStatement,ResultSet)
4. 关闭数据库操作以及连接(直接关闭连接就够了)

**DriverManager类中**:

取得数据库连接的对象: public static Connection getConnection(String url, String user,String password) throws SQLException

**Connection接口中**

数据库关闭: public void close() throws SQLException

取得数据库操作对象: public Statement createStatement() throws SQLException

**Statement接口中:**

**数据更新:** public int executeUpdate(String sql) throws SQLException //返回更新的行数

**数据查询:**ResultSet executeQuery(String sql) throws SQLException //返回的类型和Set无关

**ResultSet接口中:(注意基本不可能使用这个接口,用preparedStatement接口)**

**向下移动指针,并判断是否有数据行:** public boolean next() throws SQLException

注意:移动完指针后就可以取出数据了

**取出数据列的内容**:getInt(字段名),getDouble(字段名),getString(字段名),getDate(字段名)

范例:基本数据库连接和关闭

|  |
| --- |
| package com.company;  import java.sql.\*;  public class test {  private static final String DBDRIVER="com.mysql.jdbc.Driver";  private static final String DBURL = "jdbc:mysql://118.89.54.105:3306/test";  private static final String USER = "lw";  private static final String PASSWORD = "linwei";  public static void main(String args[])throws Exception{  //第一步:加载数据库驱动程序,此时不需要实例化,因为会有容器自己管理  Class.forName(DBDRIVER);  //第二步:连接数据库  Connection conn = DriverManager.getConnection(DBURL, USER,PASSWORD);  System.out.print(conn);  //第四部:关闭数据库  conn.close();  }  } |

范例:数据库更新

|  |
| --- |
| public class test {  private static final String DBDRIVER="com.mysql.jdbc.Driver";  private static final String DBURL = "jdbc:mysql://118.89.54.105:3306/test";  private static final String USER = "lw";  private static final String PASSWORD = "linwei";  public static void main(String args[])throws Exception{  //第一步:加载数据库驱动程序,此时不需要实例化,因为会有容器自己管理  Class.forName(DBDRIVER);  //第二步:连接数据库  Connection conn = DriverManager.getConnection(DBURL, USER,PASSWORD);  //第三步:数据库操作  //插入操作  Statement stmt = conn.createStatement();  String sql = "insert into student(name,score)values('小明',34.5)";  int len = stmt.executeUpdate(sql);  System.out.print("更新了:"+len + "行");  //修改操作  sql = "update student set name='xiaoming' where name='小明'";  stmt.executeUpdate(sql);  //数据查询  sql = "select name,score from student";  ResultSet rs = stmt.executeQuery(sql);  while(rs.next()){  String name = rs.getString("name");  double score = rs.getDouble("score");  System.out.println("姓名:" + name + ",成绩:" + score);  }  //第四部:关闭数据库  conn.close();  }  } |

Connection

客户端

getConnection

MySQL连接实现

Oracle连接实现

DriverManager

JJDBC,在驱动数据库连接对象时,采用的是工厂设计模式,而DriverManager是一个工厂类,客户端调用时会完全隐藏具体的实现子类

# 3.PreparedStatement接口

虽然JDBC里面有Statement接口,但是从实际来讲,Statement接口存在有严重的操作缺陷，不会在工作中使用.

范例:以数据增加为例来发现Statement的问题

|  |
| --- |
| public class test {  private static final String DBDRIVER="com.mysql.jdbc.Driver";  private static final String DBURL = "jdbc:mysql://118.89.54.105:3306/test";  private static final String USER = "lw";  private static final String PASSWORD = "linwei";  public static void main(String args[])throws Exception{  String name = "Mr'SMITH";  double score = 34.5;  Class.forName(DBDRIVER);  Connection conn = DriverManager.getConnection(DBURL, USER,PASSWORD);  Statement stmt = conn.createStatement();  // String sql = "insert into student(name,score)values('小明',34.5)";  String sql = "insert into student(name,score)values('"+name+"',score)";  int len = stmt.executeUpdate(sql);  //第四部:关闭数据库  conn.close();  }  } |

**问题:**

1)用变量的时候要拼接字符串,很麻烦

2)注意将报错,由于SQL语句中单引号的问题，

**而PreparedStatement和Statement不同在于前者是执行一条具有特殊占位标记的SQL语句,并且可以动态的设置所需要的数据**

PreparedStatement是Statement的子接口,获取这个接口的实例,是用

Connection接口里的: PreparedStatement prepareStatement(String sql) throws SQLException

参数sql是一个具有特殊标记的完整SQL,但是此时没有内容,通过PreparedStatement里的setXXX方法来进行设置SQL内容,并有两个方法

更新操作: int executeUpdate() throws SQLException

查询操作: ResultSet executeQuery() throws SQLException

注意这两个方法没有参数

范例:改进数据增加

|  |
| --- |
| public class test {  private static final String DBDRIVER="com.mysql.jdbc.Driver";  private static final String DBURL = "jdbc:mysql://118.89.54.105:3306/test";  private static final String USER = "lw";  private static final String PASSWORD = "linwei";  public static void main(String args[])throws Exception{  String name = "Mr'SMITH";  double score = 34.5;  Class.forName(DBDRIVER);  Connection conn = DriverManager.getConnection(DBURL, USER,PASSWORD);  String sql = "insert into student(name,score)values(?,?)";  PreparedStatement stmt = conn.prepareStatement(sql);  stmt.setString(1,name);  stmt.setDouble(2,45.6);  int len = stmt.executeUpdate();  //第四部:关闭数据库  System.out.print(len);  conn.close();  }  } |

注意:java.util.Date类有三个子类都是在java.sql包中的:

Java.sql.Date:描述的是日期

Java.sql.Time:描述的是时间

Java.sql.Timestamp:描述的是时间戳(日期时间)

如果要将java.util.Date变为java.sql.Date(Time,Timestamp),只能依靠long完成,通过java.util.Date:public long getTime():将Date变为long

Java.sql.Date:public Date(long date):将long变为sql.Date

范例:查询全部数据:

|  |
| --- |
| public class test {  private static final String DBDRIVER="com.mysql.jdbc.Driver";  private static final String DBURL = "jdbc:mysql://118.89.54.105:3306/test";  private static final String USER = "lw";  private static final String PASSWORD = "linwei";  public static void main(String args[])throws Exception{  String name = "Mr'SMITH";  double score = 34.5;  Class.forName(DBDRIVER);  Connection conn = DriverManager.getConnection(DBURL, USER,PASSWORD);  String sql = "select name,score from student where name=?";  PreparedStatement stmt = conn.prepareStatement(sql);  stmt.setString(1,name);  ResultSet rs = stmt.executeQuery();  while(rs.next()){  String x = rs.getString("name");  double y = rs.getDouble("score");  System.out.print("姓名:" + x + ",价格:"+y);  }  conn.close();  }  } |

# 4.批处理和事务处理

批处理指的是一次性向数据库之中发出多条操作命令一起执行，要想操作批处理需要使用

Statement和PreparedStatement上的方法

**Statement接口定义的方法:**

增加批处理语句: public void addBatch(String sql) throws SQLException

执行批处理: int[] executeBatch() throws SQLException//返回的数组是包含了所有批处理语句的执行结果，第一个批处理语句对应数组下标0的结果，排序排序

**PreparedStatement接口定义的方法:**

**增加批处理:** public void addBatch() throws SQLException

范例:

|  |
| --- |
| public static void main(String args[])throws Exception{  Class.forName(DBDRIVER);  Connection conn = DriverManager.getConnection(DBURL, USER,PASSWORD);  Statement stmt = conn.createStatement();  stmt.addBatch("INSERT INTO student(name,score) VALUES ('小明',34.5)");  stmt.addBatch("INSERT INTO student(name,score) VALUES ('小红',35.5)");  stmt.addBatch("INSERT INTO student(name,score) VALUES ('小化',36.5)");  stmt.addBatch("INSERT INTO student(name,score) VALUES ('小李',38.5)");  stmt.addBatch("INSERT INTO student(name,score) VALUES ('小话',39.5)");  int result[] = stmt.executeBatch();  System.out.print(Arrays.toString(result));  conn.close();  } |

由于JDBC具备自动提交的功能,如果某条sql语句错误,这条错误的语句不会执行,不是错误的语句都能执行,这显然是有问题的

JDBC提供了事务处理操作来进行手工的事务控制,所有的方法都在Connection接口中：

1. 事务提交: public void commit() throws SQLException
2. 事务回滚: public void rollback() throws SQLException
3. 设置是否为自动提交: public void setAutoCommit(boolean autoCommit)

throws SQLException

|  |
| --- |
| public static void main(String args[])throws Exception{  Class.forName(DBDRIVER);  Connection conn = DriverManager.getConnection(DBURL, USER,PASSWORD);  Statement stmt = conn.createStatement();  conn.setAutoCommit(false); //不自动提交  try {  stmt.addBatch("INSERT INTO student(name,score) VALUES ('小明',34.5)");  stmt.addBatch("INSERT INTO student(name,score) VALUES ('小红',35.5)");  // stmt.addBatch("INSERT INTO student(name,score) VALUES ('小化'',36.5)");  stmt.addBatch("INSERT INTO student(name,score) VALUES ('小李',38.5)");  stmt.addBatch("INSERT INTO student(name,score) VALUES ('小话',39.5)");  int result[] = stmt.executeBatch();  System.out.print(Arrays.toString(result));  conn.commit(); //如果没有错误会执行到这一行,进行提交  }catch (Exception e){  e.printStackTrace();  // conn.rollback();//如果出现异常，则进行回滚,其实写不写无所谓,反正不是自动提交  }  conn.close();  } |

不过以上的事务处理意义不大,一般由容器来进行处理