# 第 9 章 使用 JDBC 访问数据库

JDBC ( Java Database Connectivity) 是一种用于访问数据库和执行 SQL 语句的 Java 类库，JDBC 类库由一组 Java 类和接口组成，为 Java 开发者使用数据库提供了统一的编程接口，使得开发人员可以用纯 Java 的方式来连接数据库，并进行操作。

## 9.1 Connection 和 Statement 对象

访问数据库的第一步是和数据库源建立连接，只有建立了连接，才有可能实现在数据库和应用程序之间移动数据。

Java 提供了 Connection 连接对象，可以使用连接对象创建和管理应用程序和数据库之间的连接。

Connection 对象表示与数据源的一个唯一的会话。对于客户端/服务器数据库系统，它相当于到服务器的网络连接。

如果你开发数据库应用程序，除了必须建立和数据库源的连接之外，为了获取或修改数据库的数据，你需要使用 Statement 对象。

### 9.1.1 Connection 对象

Connection 是用来表示数据库连接的对象，对数据库的一切操作都是在这个连接的基础上进行的。

**1.** 创建数据源

要创建连接，必须要有数据源，我们可使用下面的方法创建数据源。

1）在运行服务器的计算机上，打开“控制面板”。双击“ODBC”图标，

2）单击“系统 DSN”标签，然后单击“添加”按钮。

3）从列表中选择“SQL Server”，然后单击“完成”按钮。

4）在“名称”框中键入数据源的名称（如图 10.4）。这个名称将被用来在 Java 程序中用来建立数据库的连接。

5）可以在“描述”文本框中输入注释信息。

6）在“服务器”框中输入数据库服务器的名称或 IP 地址（如图 10.4）。单击“下一步”。


图 10.4 创建数据源对话框 1

7）选择“使用用户输入登录 ID 和密码的 SQL Server 验证”单选择按钮（如图 10.5 所示）。

8）在“登录 ID”框中输入登录 ID，在密码框中输入密码（如图 10.5 所示）。单击“下

一步”按钮。


图 10.5 创建数据源对话框 2

9）图 10.6 所示的对话框中选择选择“更改默认的数据库为”复选框。在其下的下拉列表中选择要访问的数据库。单击“下一步” 。


图 10.6 创建数据源对话框 3

10）最后单击“完成”按钮。这样创建了一个在 Java 程序中可访问的数据源。

**2。加载 JDBC 驱动程序**

通过 JDBC 可将 Java 程序连接到 SQL Sever、Oracle、Sybase、Informix 等关系型数据库和其它数据源。通过将驱动程序用作到数据源的桥梁，你可以直接在 Java 中存储和检索数据。



要与数据库连接，需要 JDBC 驱动程序。可用如下方法加载 JDBC 驱动程序。

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

当创建了数据源并加载了 JDBC 驱动程序后，就可创建数据库的连接 Connection 对象。下面的实例演示了如何创建并打开到数据库的连接。

当创建了数据源并加载了 JDBC 驱动程序后，就可创建数据库的连接 Connection 对象。下面的实例演示了如何创建并打开到数据库的连接。

## 任务 9.1 创建和打开一个到 SQL Server 的连接

主要知识点：创建到数据库的连接。关闭到数据库的连接。

问题描述：创建图 10.7 所示的应用程序， 单击“连接”按钮创建并打开一个到 SQL Server 的连接。

```
package servlets;

import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;
import java.io.*;
public class ExcuteSQL    extends HttpServlet {
      public void doGet ( HttpServletRequest req, HttpServletResponse resp )
      throws ServletException,IOException
{
            try
            {
                  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
              Connection aConnection=
DriverManager.getConnection("jdbc:odbc:mydata","sa","1234");
               req.setAttribute("Message"," 连结成功…… <br>");
                  aConnection.close();
            }
            catch(ClassNotFoundException e1){
                  System.out.print(" 加载驱动器有错误 :"+e1.getMessage());
                  return;

            }
            catch(SQLException e2){
                  System.out.print(" 创建连接有错误 :"+e2.getMessage());
                  return;
            }
            RequestDispatcher dispatcher=req.getRequestDispatcher("/SQLTest.jsp");
            if(dispatcher!=null)
                dispatcher.forward(req,resp);
}
public void doPost ( HttpServletRequest req, HttpServletResponse resp )
      throws ServletException,IOException
{
      doGet(req, resp);
```

```
    }


    }
```

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4"
    xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">

   <servlet>
       <servlet-name>ExcuteSQL</servlet-name>
     <servlet-class>servlets.ExcuteSQL</servlet-class>
   </servlet>

     <servlet-mapping>
         <servlet-name>ExcuteSQL</servlet-name>
         <url-pattern>/excuteSQL</url-pattern>
       </servlet-mapping>

    </web-app>
```

SQLTest.jsp

```
<%@ page  contentType  ="text/html; charset=gb2312"  %>
<HTML>
<HEAD>
</HEAD>
  <body >
    <form action="excuteSQL" method="post" >
    <%
        java.lang.String
aMessage=(java.lang.String)  request  .getAttribute(  "Message"  );
        if  (aMessage!=  null  )
        out  .print(aMessage+  "<br>"  );
        %>
        <p>
        输入 SQL：
        </p>
```

```
            <p>
            <textarea name="txtSQL" cols="30" rows="5"></textarea>
            </p>
            <p>     
              <input  type="submit" name="Submit1" value="                连接
" >    
              <input  type="reset"  name="Reset1" value="                重置 ">
            </p>
        </form>
    </body>
    </HTML>
```

## 代码分析与讨论

**1**）加载 **JDBC** 驱动程序 。 所有使用 ODBC（开放数据库连接性，它是访问数据库的通用接口）数据源的程序都要用到 sun.jdbc.odbc.JdbcOdbcDriver —— Java 2 中一个 JDBC-ODBC 桥驱动程序。在建立到数据源的连接前，必须加载该驱动程序。在 doGet 方法中，首先装载驱动程序：

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

Class 类位于 java.lang 包中，方法 forName 装载字符串参数指定的类，并可能引发 ClassNotFoundException 异常。

JDBC-ODBC 桥产品经由 ODBC 驱动程序提供 JDBC 访问。

**2**）创建并打开连接。 装载驱动程序后，可以使用 java.sql 包中的 DriverManager 类的 getConnection 方法来建立到数据源的连接。 在 doGet 中，加载驱动程序后调用 DriverManager 类的 getConnection 方法创建一个 Connection 对象 aConnection：

```
Connection aConnection=
        DriverManager.getConnection("jdbc:odbc:mydata","sa","1234");
```



如果使用数据源时发生了错误，方法 getConnection 以及数据源的其它所有方法都可能引发 SQLException 异常。

**3**） **DriverManager.getConnection**（**String url,String user,String password**）方法。DriverManager.getConnection 方法用于建立到数据库的连接，它返回一个 Connection 对象。参数：

url：指定数据源和用于连接到该数据源的数据库连接类型。其格式如下：

jdbc:<subprotocol>:<subname>

JDBC url 的三个部分如下：

jdbc: 协议。 JDBC url 中的协议总是 jdbc。

<subprotocol> ：驱动程序或连接机制的名称，可以有一个或多个驱动程序支持。

<subname>：数据库的唯一标识符。

例如，通过 JDBC-ODBC 桥和数据库数据库的标识符 mydata 访问 northwind 数据库的 url 为：

"jdbc:odbc:mydata"

user：为用户名，其值为数据库登录帐户。

Password:为密码，其值为数据库帐户登录的密码。

仅当数据源通过用户名和密码进行保护的情况下，　　才需要后面两个参数，　否则，这两个参数可设置为空字符串（　""）。

**3**）关闭连接。 每次使用完　Connection　后都必须将其关闭。这可以使用　　Connection　对象的　Close　方法来实现。

```
aConnection.close();
```

## 9.1.2　Statement　对象

Statement对象用来执行要对数据库执行操作的一个　　SQL　语句和获得　SQL　语句产生的结果，利用　Statement 对象可直接对数据库进行处理。

### 任务 9.2：　操作数据库

主要知识点：创建　Statement 对象。使用　Statement 对象执行　SQL 命令。

问题描述：继续完成“任务　9.1：创建和打开一个到　SQL Server 的连接”，单击“执行SQL”按钮，使用文本框中输入的　　SQL 命令修改　SQL Server store 数据库。

分析：为了使用　SQL 命令修改　SQL Server store 数据库，必须创建　Statement 对象，使用 Statement 对象执行　SQL 命令操作数据库。

```
package servlets;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;
import java.io.*;
public class ExcuteSQL  extends HttpServlet {
   public void doGet ( HttpServletRequest req, HttpServletResponse
resp )
t hrows ServletException,IOException
{
      try
       {
        String sqlString=req.getParameter("txtSQL");
             Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
         Connection aConnection=
DriverManager.getConnection("jdbc:odbc:mydata","sa","1234");
          Statement aStatement=aConnection.createStatement();
       int result=aStatement.executeUpdate(sqlString);
          req.setAttribute("Message","                    影 响 数 据 库 的 行 数 为 ：
```

```
"+String.valueOf(result)+"<br>");
                aConnection.close();
            }
            catch(ClassNotFoundException e1){
              System.out.print("          加载驱动器有错误  :"+e1.getMessage());
              return;
            }
            catch(SQLException e2){
              System.out.print("          创建连接有错误  :"+e2.getMessage());
              return;
          }
            RequestDispatcher
dispatcher=req.getRequestDispatcher("/SQLTest.jsp");
            if(dispatcher!=null)
              dispatcher.forward(req,resp);
      }
      public      void      doPost      (    HttpServletRequest          req,      HttpServletResponse
resp )
      t hrows ServletException,IOException
      {
        doGet(req, resp);
      }
}
```

## 代码分析与讨论

**1**）创建 **Statement** 对象。 Statement 为一接口， 它本身不能被实例化。 但可用 Connection 对象的 createStatement 方法创建一 Statement 对象。

如下突出显示的代码调用 Connection 对象 aConnection 的 createStatement 方法，该方法返回一 Statement 对象，将其赋给 Statement 类型的对象变量 aStatement：

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection aConnection=
 DriverManager.getConnection("jdbc:odbc:mydata","sa","1234");
Statement aStatement=aConnection.createStatement();
```

**2**）使用 **Statement** 对象修改数据库。 有了 Statement 对象后，可以调用 Statement 对象的 executeUpdate 方法执行参数指定的 SQLINSERT或 UPDATE或 DELETE语句修改数据库。

如下突出显示的代码调用 Statement 对象 aStatement 的 executeUpdate 方法 的参数 sqlString 指定的 SQL命令（ sqlString 的值 来自文本区 txtSQL ）。

```
String sqlString=req.getParameter("txtSQL");
result=aStatement.executeUpdate(sqlString);
req.setAttribute("Message","          影 响 数 据 库 的 行 数 为 ：
"+String.valueOf(result)+"<br>");
```

**3**）**Statement.executeUpdate**（**String sqlString**）方法 。对 Connection 执行 SQL 命令如

INSERT、DELELE、UPDATE 和 SET 等命令。对于 UPDATE、INSERT 和 DELETE 语句，返回值为该命令所影响的行数。对于其他所有类型的命令（如 CREATE TABLE），返回值为 0。

参数 sqlString ：其值为 SQL命令（如 INSERT、UPDATE或 DELETE）字符串。

返回值：方法的返回值为执行 SQL命令后数据库受影响的行的数目。

Update products set modelnumber= 'RU0070' where productid=355

## 9.5 使用 ResultSet

ResultSet 提供了一种读取通过在数据源执行查询命令获得的结果集中的数据的一种方法。ResultSet 是一接口， 若要创建 ResultSet 对象，必须调用 Statement 对象的 executeQuery 方法，而不直接使用构造函数。 ResultSet 是一个包含表格式形式（即行和列）的查询结果集。可通过 ResultSet 对象的方法访问 ResultSet 中的数据。

### 任务 9.3 类别信息

问题描述： 对 SQL Server 中的 Store 数据库创建一数据库应用程序， 该应用程序能够实现浏览每一产品类别的相关信息（如图 10.10）。

```
<%@ page import ="java.sql.*" %>
<%@ page contentType ="text/html;charset=GB2312" %>
<table>
  <tr>
    <td> CategoryID </td>
    <td> CategoryName </td>
  </tr>
<% Connection con = null ;
   Statement stmt = null ;
```

```
        ResultSet rs =              null    ;
        try    {
          Class.forName(              "sun.jdbc.odbc.JdbcOdbcDriver"                    );
          con =
DriverManager.getConnection(                  "jdbc:odbc:mydata"           , "sa"    , "1234"    );
          stmt = con.createStatement();
          String query =                  "SELECT * FROM Categories"             ;
          rs = stmt.executeQuery(query);
            while    (rs.next()) {              %>
              <tr>
              <td>  <%=rs.getInt(          "CategoryID"        ) %></td>
              <td>  <%=rs.getString(          "CategoryName"       ) %></td>
            </tr>
  <%}
     stmt.close();
        con.close();
     }
      catch    (SQLException e) {
            out   .println(         " 数据库操作失败，产生异常：        " +e.getMessage());
     }
     finally       {
                  if   (stmt !=         null    )
                stmt.close();
                  if   (con !=        null    )
                con.close();
        }
%>
</table>
```

## 代码分析与讨论

**1）ResultSet** 的作用。 JDBC 通过 Statement 对象执行 SQL 语句可直接对数据库进行处理。当 Statement 对象执行 SQL 查询语句之后，会返回一个 ResultSet 对象，通常称为结果集。该结果集是由满足查询条件的所有行组成的集合， 它排列成表的形式， 可通过 ResultSet 对象的方法访问数据。

**2）创建 ResultSet** 对象。ResultSet 是一接口，若要创建 ResultSet 对象，必须调用 Statement 对象的 executeQuery 方法，而不直接使用构造函数。 executeQuery 方法的返回值为 ResultSet 对象。 ResultSet 对象是一个包含表格式形式（即行和列）的查询结果集。可通过 ResultSet 对象的方法访问 ResultSet 中的数据。 例如，以下代码实现创建一 ResultSet 对象 rs，

然后调用 ResultSet 对象的 getString 方法获取结果集的 CategoryName 列的值。

```
ResultSet rs=aStatement.executeQuery("SELECT * FROM Categories ");
string CateName = rs.getString("CategoryName");
```

ResultSet.getXXX 方法在给定列名称的情况下或在给定列序号的情况下，获取指定列的值。如 rs.getString("CategoryName") 或 rs.getString(2) 获取 CategoryName 列的值。

**3）ResultSet** 应用举例。查询 Categories 数据库表中的所有记录，将每条记录显示在浏览器。

以下代码首先创建一与数据库的一个连接实例 con，并通过调用该连接实例的 createStatement方法创建 Statement对象。然后，调用 Statement对象的 executeQuery方法执行参数指定的查询命令，检索所有雇员的属性并将它们存储在 ResultSet实例中。

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
con = DriverManager.getConnection("jdbc:odbc:mydata","sa","1234");
stmt = con.createStatement();
String query = "SELECT * FROM Categories";
rs = stmt.executeQuery(query);
```

调用 ResultSet.next 方法使游标移动到下一行 （我们访问的是游标指向的行） 。如果存在数据行，则 next 方法返回值为 true；否则为 false。执行 while 循环迭代 ResultSet 实例的多条记录。每一个迭代调用 ResultSet.getXXX 方法以获取一个类别的类别 ID（rs.getInt("CategoryID") ）和类别名称（ rs.getString("CategoryName") ）。

注意：调用 Statement对象的 executeQuery 方法返回的结果集中游标的最初位置在第一行的前面，因此要访问结果集需要首先调用 ResultSet.next 方法。

**4）检索 ResultSetd** 中的数据。

**ResultSetd.getXXX** 方法 。ResultSet.getXXX 方法在给定列名称的情况下或在给定列序号的情况下，获取指定列的值。如 rs.getInt("CategoryID") 或 rs.getInt(1) 获取 CategoryID 列的值。

get 方法中的 XXX 为指定数据列的数据类型在 Java 中对应的数据类型。 ResultSet 对象常用的 get 方法列在表 10.1 中。

表 10.1 SQL 数据类型和它们在 Java 中对应的数据类型

| SQL 类型（ JDBC 类型） | ResultSet 中的转换方法 | Java 类型 |
|---|---|---|
| CHAR,VARCHAR,LONGVARCHAR | getString(String columnName) | String |
| NUMERIC,DECIMAL | getBigDecimal( String columnName,int scale) | BigDecima |
| BIT,BOOLEAN | getBoolean(String columnName) | boolean |
| TINYINT | getByte(String columnName) | byte |
| SMALLINT | getShort(String columnName) | Short |
| INTEGER | getInt(String columnName) | int |
| BIGINT | getLong(String columnName) | long |
| REAL | getFloat(String columnName) | float |
| FLOAT,DOUBLE | getDouble(String columnName) | double |
| BINARY,VARBINARY LONGVARBINARY | getBytes(String columnName) | byte[] |
| DATE | getDate(String columnName) | Date |
| TIME | getTime(String columnName) | Time |

| TIMESTAMP | getTimestamp( String columnName) | Timestamp |
|---|---|---|
| CLOB | getClob(String columnName) | CLOB |
| BLOB | getBlob(String columnName) | Blob |
| ARRAY | getArray(String columnName) | Array |

**5）ResultSet.wasNull()**。ResultSet.wasNull() 方法检查该方法的前一条语句读取指定的列中是否包含不存在的或缺少的值。 如果指定的列值为空列值， 则返回为 true；否则为 false。

在调用 get 方法（例如 rs.getString("CategoryName") 等）获取指定列的值之后调用此方法来检查空列值，以避免引发错误。

如以下代码段在获取 CategoryName 列的值 rs.getString("CategoryName") 后，检查该列的值 rs.getString("CategoryName") 是否为空，如果不为空，就执行语句 out.print(CateName); 以避免发生错误。

```
string CateName = rs.getString("CategoryName");
if(!rs.wasNull() )
    out.print(CateName);
```

**6）Statement.executeQuery** 方法。 Statement.executeQuery 方法执行返回行的命令。它将 SQL 查询命令 发送到 Connection 并生成一个 ResultSet，返回值为一个 ResultSet 对象。因此调用 Statement.executeQuery 方法前必须先创建 Connection 对象和 Statement 对象。

**7）ResultSet.next** 方法： 该方法使 ResultSet 前进到下一条记录（即游标移到下一行） 。如果存在行返回值为 true；否则为 false。

ResultSet 的游标默认位置在第一条记录前面， 因此， 必须调用 next 来开始访问任何数据。

**8）**执行返回结果集的命令的步骤：

（1）创建 Connection 对象。

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection aConnection=DriverManager.getConnection(url,"sa","1234");
```

（2）创建 Statement 对象。

```
Statement aStatement=aConnection.createStatement();
```

（3）声明一个 ResultSet 变量。

```
ResultSet rs ;
```

（4）创建要执行的查询的 SQL 字符串。

```
String query = "SELECT * FROM Categories";
```

（5）调用 Statement 对象的 executeQuery 方法，将结果赋给到第 （3）步中创建的 ResultSet 变量。

```
rs=aStatement.executeQuery(query);
```

（6）使用 ResultSet 对象的 next 方法依次通过该对象，从结果集提取各个记录，直到该方法返回假。

```
while(rs.next()){
  int CategoryID =rs.getInt("CategoryID");
String CategoryName= rs.getString("CategoryName");
    ......
}
```

（7）关闭 ResultSet 和 Statement，关闭连接。

```
rs.close();
aStatement.close();
aConnection.close();
```

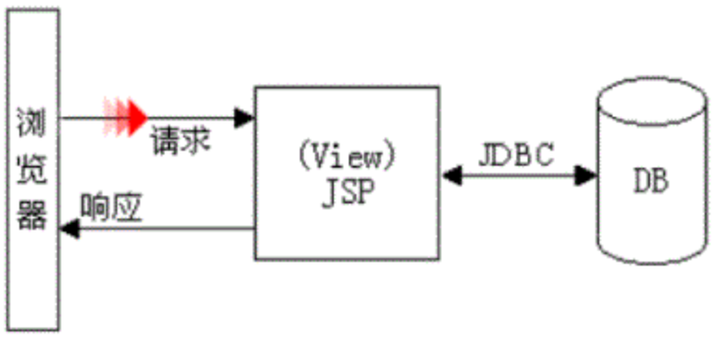
## 9.6 使用 MVC 体系结构编程

### 9.6.1 非 MVC 模式（ Model 1 ）

在 Model 1 体系中， JSP 页面独自响应请求并将处理结果返回客户。有两种结构：

。仅有 JSP 页面组成的 Web 应用程序，如图所示。

。JSP+JavaBean 技术组成的应用程序，如图所示。




1. Model 1 的特点

最直观的使用 JSP 的方法就是通过嵌入 Java 和一些特殊标签（ tag）来达到动态 HTML 的效果，仅仅由 JSP 页面构建的 Web 应用程序结果如图所示，这是最直观的使用 JSP 的方法，通过在 JSP 页面中嵌入 Java 和一些标签来达到动态 HTML 的效果，早期的 ASP 和 PHO 都是这种结构。

这种结构的优点是简单， 可以快速的搭建原型， 适合涉及几个 JSP 页面的非常小型的使用。

这种的结构缺点很多：

。HTML 和 Java 强耦合在一起。 JSP 页面中 HTML 与大量的 Java 代码交织在一起，给页面设计带来极大的困难的同时，也给阅读代码理解程序带来干扰。

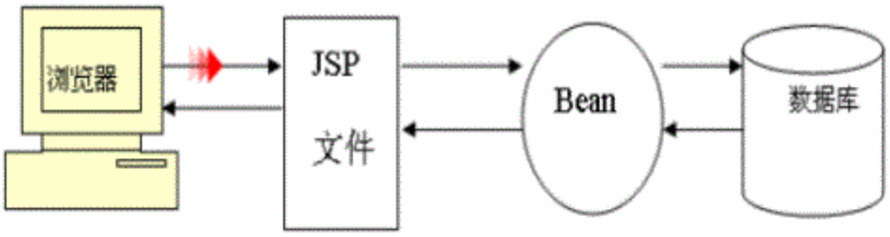
。极难维护与扩展，从图中可以看出，在 JSP 页面中直接嵌入访问数据库的代码以及 SQL 语句，会使得数据库的任何改动，都必须打开所有的 JSP 页面进行修改，当有几十个甚至几百个 JSP 页面时，改动的工作量是非常惊人的。

。不方便调试。业务逻辑与 HTML 代码， 甚至 JavaScript 代码强耦合在一起， 极难定位错误，如果以前编写过 ASP 程序，肯定有过这种痛苦的经历。

现在网上很多开源的 JSP 代码都是这种机构，本书强烈反对在实际项目中使用这种结构，这种结构完全没有体现出 JSP 技术的强大优势。

JSP+JavaBean 技术构建的 Web 应用程序如图所示。这种结构相比纯粹由 JSP 组成的应用程序结构有了很大的改进，充分利用了 Java 面向对象语言的优点，有人甚至为这种结构

起名为 Model 1.5 。




从图可以看出，业务逻辑和数据库操作从 JSP 页面中分离出来，封装在 JavaBean 中。这样就体现出众多优点：

。纯净的 JSP 页面，因为业务逻辑和数据库操作已经从 JSP 页面剥离出来， JSP 页面中只需要嵌入少量的 Java 代码甚至不使用 Java 代码。

。可重用的组件，设计良好的 JavaBean 可以重用，甚至可以作为产品销售，在团队协作的项目中，可重用的 JavaBean 将会大大减少开发人员的工作量，加快开发进度。

。方便进行调试，因为复杂的操作都封装在一个或者数个 JavaBean 中，错误比较容易定位。

。易维护扩展，系统的升级或者更改往往集中在一组 JavaBean 中，而不用编辑所有的 JSP 页面。

JSP+JavaBean 技术虽然有很多优点，当前在中小型项目中比较流行，但这种技术也有很多限制，下节将讨论该技术的使用范围。

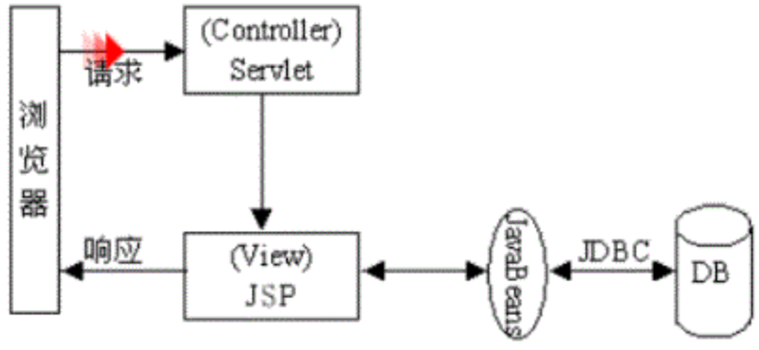
2. Model 1 的应用范围

Model 1 中纯粹使用 JSP 的结构不推荐使用，书中以后谈到的 Model 1 指的是 JSP+JavaBean构建的应用程序。

JSP+JavaBean 技术在一定程度上分离了显示（ JSP 页面）与逻辑处理（ JavaBean），使用起来也很方便，但如果项目需要在所有的请求（ Request）被处理之前进行一次统一的处理，诸如设计编码或用户权限验证等， 则在每个 JSP 页面都要加入流程控制代码， 以后对这些流程代码的修改不得不打开所有的 JSP 页面进行修改。

JSP+JavaBean 适用不需要专门流程控制的中小型项目，如果项目比较复杂，就需要借助 Model 2 了，应用 MVC 模式的 Model 2 有专门的流程控制，所以在项目复杂的情况下也可以有清晰的结构。

9.6.2 MVC 模式（ Model 2 ）



```
package components;
public class Category    {

        private int id;
        private String name;

        public Category(int id, String name) {
                this.id = id;
                this.name = name;
        }
        public Category() {}
        public int getId() {
                return id ;
        }
        public String getName() {
                return name;
        }

}
```

```
package components;

import java.sql.*;
import java.util.*;
public class CatalogDAO {
    private Connection dbConnection;
     public CatalogDAO()     {

     }
     private void getDBConnection(){
                try
                {
                        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                        dbConnection=DriverManager.getConnection("jdbc:odbc:rr","sa","1234");
                }
                catch(ClassNotFoundException e1){

                }
                catch(SQLException e2){
```

```
            }
        }

    public ArrayList getCategories(){
        String qstr = "select CategoryID, CategoryName from Categories ";
        ArrayList al = new ArrayList();
        Statement stmt = null;
            ResultSet rs = null;
            try {
                getDBConnection();
                stmt = dbConnection.createStatement();
                rs = stmt.executeQuery(qstr);
                while (rs.next()) {
                    int catid = rs.getInt("CategoryID");
                    String name = rs.getString("CategoryName").trim();
                    Category    cat = new Category(catid, name);
                    al.add(cat);
                }
            } catch(SQLException se) {
                System.out.println("SQLException while getting " +
                        "multiple categories : " + se.getMessage());
            } finally {

            }
            return al;
    }
}



  package dispatcher;

  import javax.servlet.*;
  import javax.servlet.http.*;
  import components.*;
  import java.sql.*;
  import java.io.*;
  import java.util.*;
  public class Dispatcher extends HttpServlet {
        public void doGet(HttpServletRequest req, HttpServletResponse res)
            throws ServletException, IOException {
            HttpSession session = req.getSession(true);
            //String selectedScreen = req.getServletPath();
            CatalogDAO catalogDAO = (CatalogDAO)
```

```
session.getAttribute("catalogDAO");
                    if (catalogDAO == null) {
                        catalogDAO = new CatalogDAO();
                        session.setAttribute("catalogDAO", catalogDAO);
                    }
                            ArrayList cat = new ArrayList();
        cat=catalogDAO. getCategories ();
                            req.setAttribute("Categories",cat);
                    RequestDispatcher rd = req.getRequestDispatcher("/getCategories.jsp");
                    rd.forward(req, res);

            }
               public void doPost(HttpServletRequest req, HttpServletResponse res)
                  throws ServletException, IOException {
                  doGet(req, res);
               }
            }
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
   http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd" >
   <servlet >
```

```
<servlet-name >Dispatcher </ servl
et-name >

<servlet-class >dispatcher.Dispa
tcher </ servlet-class >
</ servlet >
<servlet-mapping >

<servlet-name >Dispatcher </ servl
et-name >

<url-pattern >/getCategories </ ur
l-pattern >
</ servlet-mapping >

</ web-app >
```



getCategories.jsp

```
<%@page contentType ="text/html;
charset=gb2312" language ="java"
import ="java.util.*"
```

```
errorPage     ="errorpage.jsp"            %>
<%@page   import    ="components.*"        %>
<table    >
     <tr   >
     <td  >CategoryID      </  td  >
     <td  >CategoryName    </  td  >
     </  tr   >
          <%

   if   (request.getAttribute(              "Cate
gories"      )!=   null    ) {
   ArrayList
aList=(ArrayList)request.getAtt
ribute(      "Categories"         );
       for    (   int
i=0;i<aList.size();i++)
   {
    Category  cat =(Category)
aList.get(i);
   %>
              <tr   >
```

```
<td ><%=cat.getId()          %></ td >

<td ><%=cat.getName()        %></ td >
          </ tr  >
          <%
 }}
   %>
      </ table    >
```

```
package dispatcher;
import javax.servlet.*;
import javax.servlet.http.*;
import components.*;
import java.sql.*;
import java.io.*;
public class Dispatcher2 extends HttpServlet {
                private String username ;
                private String userpassword ;
                HttpSession session；
               ArrayList cat = new ArrayList()  ；
public void init(ServletConfig config) throws ServletException {
      super.init(config);
      username = getInitParameter("def_username");
      userpassword = getInitParameter("def_password");
}

      public void doGet(HttpServletRequest req, HttpServletResponse res)
            throws ServletException, IOException {
            session = req.getSession(true);
            String selectedScreen = req.getServletPath();
            CatalogDAO catalogDAO = (CatalogDAO) session.getAttribute("catalogDAO");
            if (catalogDAO == null) {
                  catalogDAO = new CatalogDAO();
```

```
            session.setAttribute("catalogDAO", catalogDAO);
        }
        if (selectedScreen.equals("/getCategories2 ")) {
        if(session.getAttribute("isLoggedIn")==null
                ||   !Boolean.parseBoolean(session.getAttribute("isLoggedIn").toString()))
                    selectedScreen ="/login.jsp");
            else
                    selectedScreen="/getCategories2.jsp";
        }
        else if (selectedScreen.equals("/login")) {
                String name = request.getParameter("Name");
                String password = request.getParameter("Password");
               if(name.equals(username) && password.equals(userpassword)) {
                            boolean isLoggedIn=true;
                            session.setAttribute("isLoggedIn",String.valueOf(isLoggedIn));
                            cat=catalogDAO. getCategories ();
                            session.setAttribute("Categories",cat);
                            selectedScreen="/getCategories2.jsp";
                    }
              else {
      session.setAttribute("Message"," 你输入的用户名或密码不正确     ,请重新输入  <br>");
                    selectedScreen="/ login.jsp";
                }
        }

        RequestDispatcher rd = getServletContext().getRequestDispatcher(selectedScreen);
        rd.forward(req, res);

    }
      public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        doGet(req, res);
      }
    }
```

```
    <servlet>
      <servlet-name>Dispatcher2</servlet-name>
    <servlet-class>dispatcher.Dispatcher2</servlet-class>
  </servlet>
      <servlet-mapping>
        <servlet-name>Dispatcher2</servlet-name>
        <url-pattern>/                    getCategories2</url-pattern>
```

```
        </servlet-mapping>
        <servlet-mapping>
            <servlet-name>Dispatcher2</servlet-name>
            <url-pattern>/login</url-pattern>
        </servlet-mapping>
```

Login.jsp

```
<%@ page contentType="text/html; charset=gb2312"  %>
<HTML>
<HEAD>
</HEAD>
  <body >
    <form action="Login" method="post" >
    <%
        if (session.getAttribute("Message")!=null) {
            java.lang.String aMessage=(java.lang.String) session.getAttribute("Message");
            out.print(aMessage+"<br>");
        }
        %>
        输入姓名：  <input type="text" input size=20 name="Name">
        <p>
        输入密码：  <input type="password"    nasize=20 name="Password">
        </p>
        <p>    
            <input   type="submit" name="Submit1" value="  提交 " >  
            <input   type="reset"   name="Reset1" value=" 重置 ">
        </p>
    </form>
  </body>
</HTML>
```

getCategories.jsp

```
<%@ page   contentType      ="text/html; charset=gb2312"          language    ="java"
import   ="java.util.*"          errorPage     ="errorpage.jsp"          %>
<%@ page   import   ="components.*"       %>
<table>
```

```
    <tr>
        <td>  CategoryID     </td>
        <td>  CategoryName     </td>
    </tr>
        <%
if(session.getAttribute("isLoggedIn")==null
||   !Boolean.parseBoolean(session.getAttribute("isLoggedIn").toString()))
            {
                RequestDispatcher dispatcher=req.getRequestDispatcher("Login.jsp");
                if(dispatcher!=null)
                    dispatcher.forward(req,resp);
            }

            if(session.getAttribute("Categories ")!=null) {
        ArrayList aList=(ArrayList) session.getAttribute("Categories");
      for   (  int    i=0;i       aList.size();i++)
 {
       Category  cat =(Category)                aList.get(i);
      %>
            <tr>
            <td>  <%=cat.getId()          %></td>
            <td>  <%=cat.getName()        %></td>
        </tr>
        <%
}}
  %>
      </table>
```

Login.jsp

```
<%@page  contentType  ="text/html; charset=gb2312"          %>
<HTML>
<HEAD>
</ HEAD>
  <body  >
      <form   action  ="login"     method="post"    >
      <%
            if   (session.getAttribute(          "Message" )!= null   ) {
       java.lang.String aMessage=(java.lang.String) session.getAttribute(                    "Message" );
            out.print(aMessage+       "<br>"  );
}
```

```
            %>
            输入姓名： <input  type ="text"  input size  =20  name="Name">
            <p>
            输入密码： <input  type ="password"  nasize  =20  name="Password" >
            </ p>
            <p>    
              <input  type ="submit"  name="Submit1"  value =" 提交 "  >  
              <input  type ="reset"  name="Reset1"  value =" 重置 " >
           </ p>
      </ form >
  </ body>
</ HTML>
```

```
package dispatcher;

import  javax.servlet.*;
import  javax.servlet.http.*;
import  components.*;
import  java.sql.*;
import  java.io.*;
import  java.util.*;
public  class  Dispatcher2  extends  HttpServlet {
private  String  username ;
                private  String  userpassword  ;
        HttpSession              session  ;
ArrayList  cat  =  new ArrayList();
    public  void  init(ServletConfig config)          throws  ServletException {
          super .init(config);
          username = getInitParameter(      "def_username"  );
          userpassword  = getInitParameter(      "def_password"  );
     }

    public  void  doGet(HttpServletRequest req, HttpServletResponse res)
        throws  ServletException, IOException {
        session  = req.getSession(  true  );
    String selectedScreen = req.getServletPath();
    CatalogDAO catalogDAO = (CatalogDAO)          session  .getAttribute(      "catalogDAO"  );
        if  (catalogDAO ==  null  ) {
```

```
        catalogDAO =              new CatalogDAO();
                session  .setAttribute(       "catalogDAO"  , catalogDAO);
    }
        if  (selectedScreen.equals(       "/getCategories2"     )) {
            if  ( session  .getAttribute(       "isLoggedIn"    )== null
|| !Boolean.      parseBoolean( session  .getAttribute(       "isLoggedIn"    ).toString()))
          selectedScreen =                    "/Login.jsp"     ;
else
        selectedScreen=                 "/getCategories2.jsp"       ;
    }
        else   if  (selectedScreen.equals(       "/login"     )) {
         String name = req.getParameter(                      "Name");
         String password = req.getParameter(                        "Password"  );
                  if  (name.equals(   username) && password.equals(     userpassword  )) {
                      boolean  isLoggedIn=   true  ;
                    session  .setAttribute(       "isLoggedIn"    ,String.    valueOf (isLoggedIn));
                      cat =catalogDAO.getCategories ();
                    session  .setAttribute(       "Categories"    , cat  );
          selectedScreen=                    "/getCategories2.jsp"       ;
           }
                else  {
                    session  .setAttribute(       "Message" , " 你输入的用户名或密码不正确     , 请重新输入
<br>" );
            selectedScreen =                    "/Login.jsp"     ;
          }
  }

    RequestDispatcher rd = getServletContext().getRequestDispatcher(selectedScreen);
    rd.forward(req, res);


  }
      public     void   doPost(HttpServletRequest req, HttpServletResponse res)
        throws   ServletException, IOException {
    doGet(req, res);
   }
  }
```

```
<servlet   >
      <servlet-name   >Dispatcher   </ servlet-name   >
            <servlet-class      >dispatcher.Dispatcher       </ servlet-class      >
```

```
    </servlet>

        <servlet>
            <servlet-name>Dispatcher2</servlet-name>
            <servlet-class>dispatcher.Dispatcher2</servlet-class>
            <init-param>
                <param-name>def_username</param-name>
                <param-value>Tom</param-value>
            </init-param>
            <init-param>
                <param-name>def_password</param-name>
                <param-value>123456</param-value>
            </init-param>

        </servlet>
            <servlet-mapping>
                <servlet-name>Dispatcher2</servlet-name>
                <url-pattern>/getCategories2</url-pattern>
            </servlet-mapping>

            <servlet-mapping>
                <servlet-name>Dispatcher</servlet-name>
                <url-pattern>/getCategories</url-pattern>
            </servlet-mapping>

            <servlet-mapping>
                <servlet-name>Dispatcher2</servlet-name>
                <url-pattern>/login</url-pattern>
            </servlet-mapping>
```



```
<%@page contentType="text/html; charset=gb2312" language="java" import="java.util.*"
errorPage="errorpage.jsp" %>
<%@page import="components.*" %>
<table>
    <tr>
        <td>CategoryID</td>
        <td>CategoryName</td>
    </tr>
        <%
if (session.getAttribute("isLoggedIn")==null
```

```
|| !Boolean.parseBoolean(session.getAttribute(          "isLoggedIn"   ).toString()))
    {
      response.sendRedirect(          "Login.jsp"    );

    }

          if (session.getAttribute(       "Categories"   )!= null ) {
 ArrayList aList=(ArrayList) session.getAttribute(                "Categories"   );
   for  ( int  i=0;i<aList.size();i++)
 {
     Category  cat =(Category) aList.get(i);
    %>
        <tr >
        <td ><%=cat.getId()    %></ td >
        <td ><%=cat.getName()  %></ td >
      </ tr >
      <%
}}
 %>
   </ table  >
```

## 任务 9-4 产品类别管理（ 1）

问题描述： 对 SQL Server 中的 Northwind 数据库创建一数据库应用程序， 该应用程序能够实现浏览每一产品类别的相关信息， 能对产品类别的相关信息进行修改、 查询，能够添加、删除产品类别（类别 ID，类别名称，类别描述）（如图 8-7）。

1）模型设计

模型包括存储业务数据的对象和访问数据对象。 由问题描述可知， 要存储的业务数据是产品类别。 因此设计一产品类别类 Category，该类的对象存储产品类别的信息。 要存储产品类别对象，首先读取产品类别的属性以获取类别 ID、类别名称和类别描述。然后将这些值写入一个类别数据库中， 以后可以按照相反的顺序重新创建该类别实例： 首先从类别数据库表中读取数据，然后使用数据库表中的数据填充属性，从而来实例化类别。

访问数据对象包含访问数据库读取产品类别、添加产品类别记录，修改产品类别记录，删除产品类别记录的方法。这些方法是：

读取所有产品类别。 getCategories 方法访问数据库，读取包含产品类别的属性值的类别记录，为每条记录创建类别对象，然后将这些类别对象添加到 ArrayList 对象 categories 中，

并将其返回。

添加类别。 addCategory 方法向 categories 数据库表中添加一条记录。 此方法以参数的形式接受新 categories 对象，而且不会返回数据。

```
public void addCategory(Category aCategory)
```

修改类别。 updateCategory 方法用参数指定的类别的属性值修改当前雇员对应的数据库表中的记录。

```
public void updateCategory(Category aCategory)
```

删除类别。 delete 方法删除参数指定的类别 ID 对应的数据库表中的记录。

```
public void deleteCategory(int categoryId)
```

```
package components;
public class Category {

    private int id;
    private String name;


    public Category(int id, String name) {
        this.id = id;
        this.name = name;
    }
    public Category(String name) {

        this.name = name;
    }

    /**
     * Class constructor with no arguments, used by the web tier.
     */
    public Category() {}

    public int getId() {
        return id ;
    }

    public String getName() {
        return name;
    }

}
```

```
package components;
import java.sql.*;
import java.util.*;
public class CatalogDAO {
        private Connection dbConnection;

         public CatalogDAO()      {

         }
         private void getDBConnection(){
                  try
                  {
                           Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

dbConnection=DriverManager.getConnection("jdbc:odbc:mydata","sa","1234");
                  }
                  catch(ClassNotFoundException e1){

                  }
                  catch(SQLException e2){

                  }
             }

         public Category getCategory(int categoryId)         {
               String qstr = "select CategoryID, CategoryName from Categories "
                    + " where CategoryID = " + categoryId ;

                  Category cat = null;
                  Statement stmt = null;
                  ResultSet rs = null;
                  try {
                        getDBConnection();
                        stmt = dbConnection.createStatement();
                        rs = stmt.executeQuery(qstr);
                        while (rs.next()) {
                              int catid = rs.getInt("CategoryID");
                              String name = rs.getString("CategoryName");
                              cat = new Category(catid, name);
                        }
                  } catch(SQLException se) {
                        System.out.println("SQLException while getting " +
                                          "Category " + categoryId + " : " + se.getMessage());
```

```
            } finally {
                closeResultSet(rs);
                closeStatement(stmt);
                closeConnection();
            }
            return cat;
}

public ArrayList getCategories(){
    String qstr = "select CategoryID, CategoryName from Categories ";
    ArrayList al = new ArrayList();
    Statement stmt = null;
        ResultSet rs = null;
        try {
            getDBConnection();
            stmt = dbConnection.createStatement();
            rs = stmt.executeQuery(qstr);
            while (rs.next()) {
                int catid = rs.getInt("CategoryID");
                String name = rs.getString("CategoryName").trim();
                Category    cat = new Category(catid, name);
                al.add(cat);
            }
        } catch(SQLException se) {
            System.out.println("SQLException while getting " +
                    "multiple categories : " + se.getMessage());
        } finally {
            closeResultSet(rs);
            closeStatement(stmt);
            closeConnection();
        }
        return al;
}

public void     addCategory(Category aCategory) {
    String qstr = "Insert Into Categories (CategoryName) Values("
        +"'"+aCategory.getName()+"')";
        Statement stmt = null;
        try {
            getDBConnection();
            stmt = dbConnection.createStatement();
            stmt.executeUpdate(qstr);
```

```
            } catch(SQLException se) {
                System.out.println("SQLException while Inserting " +
                                "Category "     + " : " + se.getMessage());
            } finally {
                closeStatement(stmt);
                closeConnection();
            }
}

public void     updateCategory(Category aCategory)    {
        String qstr = "Update Categories     set CategoryName="
              +"'"+aCategory.getName()+"'"
              + "    Where CategoryID = "+ aCategory.getId();
            Statement stmt = null;
            try {
                getDBConnection();
                stmt = dbConnection.createStatement();
                stmt.executeUpdate(qstr);

            } catch(SQLException se) {
                System.out.println("SQLException while Updating " +
                                "Category "     + " : " + se.getMessage());
            } finally {
                closeStatement(stmt);
                closeConnection();
            }
}

public void     deleteCategory(int categoryId)       {
        String qstr = "Delete Categories "
              + " Where CategoryID = " + categoryId ;
            Statement stmt = null;
            try {
                getDBConnection();
                stmt = dbConnection.createStatement();
                stmt.executeUpdate(qstr);

            } catch(SQLException se) {
                System.out.println("SQLException while deleting " +
                                "Category " + categoryId + " : " + se.getMessage());
            } finally {
                closeStatement(stmt);
                closeConnection();
```

```
                }
        }


        private void closeStatement(Statement stmt)    {
                try {
                        if (stmt != null) {
                                stmt.close();
                        }
                } catch(SQLException se) {
                        System.out.println("SQLException while closing " +
                                                "statement    : " + se.getMessage());
                }
        }

        private void closeResultSet(ResultSet result)    {
                try {
                        if (result != null) {
                                result.close();
                        }
                } catch(SQLException se) {
                        System.out.println("SQLException while closing " +
                                                "result    : " + se.getMessage());
                }
        }

        private void closeConnection()    {
                try {
                        if (dbConnection != null && !dbConnection.isClosed()) {
                                dbConnection.close();
                        }
                } catch(SQLException se) {
                        System.out.println("SQLException while closing " +
                                                "DB connection    : " + se.getMessage());
                }
        }


}



package dispatcher;
```

```
import javax.servlet.*;
import javax.servlet.http.*;
import components.*;
import java.sql.*;
import java.io.*;
public class Dispatcher extends HttpServlet {

    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {

        HttpSession session = req.getSession(true);
        String selectedScreen = req.getServletPath();
        CatalogDAO catalogDAO = (CatalogDAO) session.getAttribute("catalogDAO");
        if (catalogDAO == null) {
            catalogDAO = new CatalogDAO();
            session.setAttribute("catalogDAO", catalogDAO);
        }
        if (selectedScreen.equals("/newCategory")) {
            String CategoryName=req.getParameter("CategoryName");
            if(CategoryName==null)
            {
                req.setAttribute("action","newCategory");
                selectedScreen="/updateCategory.jsp";

            }
            else
            {
                Category aCategory = new    Category(CategoryName);
                catalogDAO.addCategory(aCategory);
                selectedScreen="/getCategories.jsp";
            }
        }

        else if (selectedScreen.equals("/updateCategory")) {
            String CategoryName=req.getParameter("CategoryName");
            if(CategoryName==null)
            {
                session.setAttribute("CategoryID",req.getParameter("CategoryID"));
                req.setAttribute("action","updateCategory");
                selectedScreen="/updateCategory.jsp";
            }
            else
            {
```

```
                    String categoryID=(String)session.getAttribute("CategoryID");
                    Category                                        aCategory=new
Category(Integer.parseInt(categoryID),CategoryName);
                    catalogDAO.updateCategory(aCategory);
                    selectedScreen="/getCategories.jsp";
                }
            }
            else if (selectedScreen.equals("/deleteCategory"))
            {
                int categoryID =Integer.parseInt(req.getParameter("CategoryID"));
                catalogDAO.deleteCategory(categoryID);
                selectedScreen="/getCategories.jsp";
            }


            RequestDispatcher rd = getServletContext().getRequestDispatcher(selectedScreen);
            rd.forward(req, res);

        }

          public void doPost(HttpServletRequest req, HttpServletResponse res)
            throws ServletException, IOException {
            doGet(req, res);

          }


        }
```

```
    <servlet>
      <servlet-name>Dispatcher</servlet-name>
    <servlet-class>dispatcher.Dispatcher</servlet-class>
  </servlet>

      <servlet-mapping>
        <servlet-name>Dispatcher</servlet-name>
        <url-pattern>/newCategory</url-pattern>
      </servlet-mapping>
      <servlet-mapping>
        <servlet-name>Dispatcher</servlet-name>
        <url-pattern>/updateCategory</url-pattern>
      </servlet-mapping>
      <servlet-mapping>
```

```
        <servlet-name>Dispatcher</servlet-name>
        <url-pattern>/deleteCategory</url-pattern>
    </servlet-mapping>
```

getCategories.jsp

```
<%@ page contentType ="text/html; charset=gb2312" language ="java"
import ="java.util.*" errorPage ="errorpage.jsp" %>
<%@ page import ="components.*" %>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title> 产品类别 </title>
</head>

<body bgcolor="#0099FF" text="#FFFFFF" link="#33FF00">

<p>   </p>
<p align="center"><font color="#00FF00" size="+3" face="华文行楷 "> 所有产
品类别 </font></p>
<p><a href="newCategory"><font size="+1" face="华文行楷 "> 新加产品类别
</font></a></p>

  <div align="center">
    <table width="75%" border="1">
      <tr>
        <td> 类别编号 </td>
        <td> 类别名称 </td>
        <td> 删除 </td>
        <td> 更新 </td>
      </tr>
      <%
  String CategoryID= "" ;
  String CategoryName= "" ;
 CatalogDAO aCatalogDAO= new CatalogDAO();
 ArrayList al =aCatalogDAO.getCategories();
    for ( int i=0;i<al.size();i++)
 {
      Category cat =(Category)al.get(i);
    %>
        <tr>
        <td> <%=cat.getId() %></td>
```

```
            <td>  <%=cat.getName()        %></td>
            <td>  <a href="deleteCategory?&CategoryID=                    <%=cat.getId()        %>">  删除
</a></td>
            <td>  <a href="updateCategory?CategoryID=                    <%=cat.getId()        %> ">  更新
</a>    </td>
          </tr>
          <%
}
  %>
      </table>
   </div>

</body>
</html>
```

updateCategory.jsp：

```
<%@ page    contentType       ="text/html; charset=gb2312"                  language     ="java"
import     ="java.sql.*"             errorPage      =""     %>
<%
 String action=(String)                  request     .getAttribute(             "action"       );
%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>        <%=action     %></title>
</head>
<body bgcolor="#0099FF" text="#FFFFFF" link="#33FF00">


<p align="center"><font color="#00FF00" size="+3" face="                                        方正舒体
"> <%=action     %></font></p>
<p align="center">                   </p>
<form  method="post" action="                  <%=action     %>">
   <table width="49%" height="50"  border="1" align="center"
cellpadding="0" cellspacing="0">
      <tr>
         <td width="48%">          类别名称  </td>
         <td width="52%"><input name="CategoryName" type="text" ></td>
      </tr>
   </table>
   <p align="center">
      <input type="submit" name="Submit" value="                              提交 ">
   </p>
```

```
</form>
<p>    </p>
<p><a href="getCategories.jsp">                    &lt;&lt;Back          </a></p>
  </body>
</html>
```

```
public class CatalogDAO {
      private Connection dbConnection;
 public CatalogDAO()     { }
       private void getDBConnection(){
                try
                {
                        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
           dbConnection=DriverManager.getConnection("jdbc:odbc:mydata","sa","1234");
                }
           catch(ClassNotFoundException e1){}
           catch(SQLException e2){ }
           }
             public Category getCategory(int categoryId)        {
             String qstr = "select CategoryID, CategoryName from Categories "
                    + " where CategoryID = " + categoryId ;
                ____________________________________________________________
                ____________________________________________________________
                ____________________________________________________________
               ____________________________________________________________
               ____________________________________________________________
               ____________________________________________________________
               ____________________________________________________________
                        while (rs.next()) {
                           ____________________________________
                           ____________________________________
                                cat = new Category(catid, name);
                        }
                    } catch(SQLException se) {
                        } finally {
                        closeResultSet(rs);
                        closeStatement(stmt);
                        closeConnection();
                    }
                    return cat;
          }
```

```
public ArrayList getCategories(){
ArrayList al = new ArrayList();
while (rs.next()) {
```

## 11.3 使用 PreparedStatement

在前面章节中，我们看到了如何使用 Statement 对象执行 SQL 命令，使用 Statement 对象执行 SQL 语句时， SQL 语句传递给 RDBMS ， RDBMS 对 SQL 进行编译，使之转换为可理解的格式。显然，这种编译过程是一种开销。对于不经常执行的 SQL 语句，这可能不算是个大问题。

为了消除重复编译 SQL 语句所产生的开销， JDBC 提供了 PreparedStatement，一个 PreparedStatement 是一个预编译的 SQL 语句，它执行时没有编译的开销。 DBMS 只是执行预备语句（ PreparedStatement），因而提供了更快的响应。

### 11.3.1 创建 PreparedStatement

与 Statement 对象一样， PreparedStatement 对象是用一个活动的 Connection 对象创建的。创建这两个对象的不同处在于，对于 PreparedStatement，必须在创建时而不是执行时指定 SQL 语句。为此，要使用 Connection 对象的 prepareStatement 方法创建 PreparedStatement 对象。下面的代码显示了如何从 Connection 对象获得 PreparedStatement 对象。该代码首先定义了一个更新 Categories 数据库表 CategoryName 字段和 Description 字段的语句，在最后一行 aConnection 对象的 prepareStatement 方法创建 PreparedStatement 对象 pstmt。

```
String strSQL = "UPDATE Categories SET";
strSQL += " CategoryName=? , ";
strSQL += " Description=? " ;
strSQL += " WHERE  CategoryID=?";
PreparedStatement pstmt=aConnection.prepareStatement(strSQL);
```

可以看到以上代码在第 2 行、3 行和四行，将类别的 CategoryID 值为？的 CategoryName 字段的值修改为“？” ，Description 字段的值修改为“？” 。 这是什么意思呢？这就是告诉 PreparedStatement 对象在运行时将提供这些值。问号（？）是在运行时提供的实际数据的占位符。 用这个 SQL 为参数调用 prepareStatement 方法时， 它准备接受运行时数据、 编译 SQL 语句并将它存储到 pstmt 变量中。

### 11.3.2 执行 PreparedStatement

创建了 PreparedStatement对象后，就可以执行它了。但是在执行它之前，必须为在创建PreparedStatement时定义的占位符提供实际的数据。可以调用 PreparedStatement接口中相应的 setXXX 方法完成这项任务。如果要替换的值是一个 int 值，那么调用接口中的 setInt 方法。基本上可以找到 Java类型的 setXXX 方法。如果要用 SQL NULL 替换占位符，则使用 setNull 方法。

让我们将存储在 pstmt 中的预备语句改为将 CategoryID 值为 1 的类别的 CategoryName 字段的值修改为“ Dairy Products”，Description 字段的值修改为“ Cheeses”，为此，使用以下的代码段：

```
pstmt.setString(1,"Dairy Products");
pstmt.setString(2,"Cheeses");
pstmt.setInt(3,1);
```

setXXX 方法的第一个参数指定要替换的占位符的索引（索引从 1 开始），pstmt.setString(1,"Dairy Products") 将第一个占位符（索引为 1）换为 "Dairy Products"，pstmt.setString(2,"Cheeses") 将第二个占位符 （索引为 2）换为 " Cheeses "，pstmt.setInt(3,1) 将第三个占位符 （索引为 3）换为 " Cheeses "，变量 pstmt 就表示以下 SQL 语句编译后的版本：

```
UPDATE Categories SET
  CategoryName ="Dairy Products"
  Description =" Cheeses "
  WHERE CategoryID=1
```

剩下的就将它发送给 DBMS 以执行这个语句。可以通过调用 PreparedStatement对象的 executeUpdate()方法做到这一点。不像执行 Statement 的情形，不需要向 executeUpdate 方法传递任何东西，下面的代码将执行这个 PreparedStatement：

```
pstmt.executeUpdate();
```

### 实例 11-4 产品类别管理（ 2）

问题描述：使用 PreparedStatement对 SQL Server 中的 Northwind 数据库创建一数据库应用程序，该应用程序能够实现浏览每一产品类别的相关信息，能对产品类别的相关信息进行修改、查询，能够添加、删除产品类别（类别 ID，类别名称，类别描述）（如图 8-7）。

分析：根据问题的描述该应用程序和产品类别管理 （1）应用程序的问题域类、 GUI（图形用户界面）类是相同的，不同的是数据存取方式不同，在这里要求使用 PreparedStatement存取数据库的数据，因此，只需重新设计数据存取类—— CategoryDA 即可，这是采用三层应用程序设计的方法的好处。

```
package category;
import java.sql.*;
import java.util.*;
public class CategoryDAO {
private Connection dbConnection;
```

```
        public CategoryDAO() {

        }

        private void getDBConnection(){
                try
                {
                  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

dbConnection=DriverManager.getConnection("jdbc:odbc:mydata","sa","123
4");
                }
                catch(ClassNotFoundException e1){

                }
                catch(SQLException e2){

              }
          }

        public Category getCategory(int categoryId)                    {
             String    qstr    =  "select    CategoryID,    CategoryName,    Description    from
Categories "
              + " where CategoryID =? ";

            Category cat = null;
            PreparedStatement pstmt = null;
            ResultSet rs = null;
            try {
               getDBConnection();
               pstmt=dbConnection.prepareStatement(qstr);
               pstmt.setInt(1,categoryId);
               rs = pstmt.executeQuery();
               while (rs.next()) {
                  int catid = rs.getInt("CategoryID");
                  String name = rs.getString("CategoryName");
                  String desc = rs.getString("Description");
                  cat = new Category(catid, name,desc);
               }
            } catch(SQLException se) {
               System.out.println("SQLException while getting " +
                     "Category                    "    +    categoryId          +    "    :    "    +
se.getMessage());
```

```
            } finally {
                closeResultSet(rs);
                closePreparedStatement(pstmt);
                closeConnection();
            }
            return cat;
        }

        public ArrayList getCategories(){
            String qstr = "select CategoryID, CategoryName , Description from
Categories ";
            ArrayList al = new ArrayList();
            PreparedStatement pstmt = null;
            ResultSet rs = null;
            try {
                getDBConnection();
                pstmt = dbConnection.prepareStatement(qstr);;
                rs = pstmt.executeQuery();
                while (rs.next()) {
                    int catid = rs.getInt("CategoryID");
                    String name = rs.getString("CategoryName").trim();
                    String desc = rs.getString("Description").trim();
                    Category cat = new Category(catid, name,desc);
                    al.add(cat);
                }
            } catch(SQLException se) {
                System.out.println("SQLException while getting " +
                        "multiple categories : " + se.getMessage());
            } finally {
                closeResultSet(rs);
                closePreparedStatement(pstmt);
                closeConnection();
            }
            return al;
        }

        public void  addCategory(Category aCategory) {
            String qstr = "INSERT INTO Categories ";
            qstr  += "(CategoryName,Description)";
            qstr  += " VALUES(?,?)";

            PreparedStatement pstmt = null;
            try {
                getDBConnection();
```

```
            pstmt = dbConnection.prepareStatement(qstr);
            pstmt.setString(1,aCategory.getCategoryName());
                  pstmt.setString(2,aCategory.getDescription());
                  pstmt.executeUpdate();

        } catch(SQLException se) {
            System.out.println("SQLException while Inserting " +
                  "Category "  + " : " + se.getMessage());
        } finally {
            closePreparedStatement(pstmt);
            closeConnection();
        }
    }

    public void  updateCategory(Category aCategory)                    {
        String qstr = "UPDATE Categories SET";
        qstr += " CategoryName=? , ";
        qstr += " Description=? " ;
        qstr += " WHERE  CategoryID=?";
        PreparedStatement pstmt = null;
        try {
            getDBConnection();
            pstmt = dbConnection.prepareStatement(qstr);
            pstmt.setString(1,aCategory.getCategoryName());
                  pstmt.setString(2,aCategory.getDescription());
                  pstmt.setInt(3,aCategory.getCategoryID());
                  pstmt.executeUpdate();

        } catch(SQLException se) {
            System.out.println("SQLException while Updating " +
                  "Category "  + " : " + se.getMessage());
        } finally {
            closePreparedStatement(pstmt);
            closeConnection();
        }
    }

    public void  deleteCategory(int categoryId)                    {
        String qstr = "Delete Categories "
         + " Where CategoryID =? "  ;
        PreparedStatement pstmt = null;
        try {
            getDBConnection();
            pstmt = dbConnection.prepareStatement(qstr);
```

```
            pstmt.setInt(1,categoryId);
            pstmt.executeUpdate();

        } catch(SQLException se) {
            System.out.println("SQLException while deleting " +
                    "Category " + categoryId + " : " +
se.getMessage());
        } finally {
            closePreparedStatement(pstmt);
            closeConnection();
        }
    }


    private void closePreparedStatement(PreparedStatement pstmt)  {
        try {
            if (pstmt != null) {
                pstmt.close();
            }
        } catch(SQLException se) {
                System.out.println("SQLException while closing " +
                        "PreparedStatement  : " + se.getMessage());
        }
    }

    private void closeResultSet(ResultSet result)  {
        try {
            if (result != null) {
                result.close();
            }
        } catch(SQLException se) {
                System.out.println("SQLException while closing " +
                        "result  : " + se.getMessage());
        }
    }

    private void closeConnection()  {
        try {
            if (dbConnection != null && !dbConnection.isClosed()) {
                dbConnection.close();
            }
        } catch(SQLException se) {
                System.out.println("SQLException while closing " +
                        "DB connection  : " + se.getMessage());
```

```
            }
        }
    }


    package category;

    import javax.servlet.*;
    import javax.servlet.http.*;
    import java.io.*;
    public class Dispatcher extends HttpServlet {

        public void doGet(HttpServletRequest req, HttpServletResponse
res)
            throws ServletException, IOException {

            HttpSession session = req.getSession(true);
            String selectedScreen = req.getServletPath();
            CatalogDAO catalogDAO = (CatalogDAO)
session.getAttribute("catalogDAO");
            if (catalogDAO == null) {
               catalogDAO = new CatalogDAO();
               session.setAttribute("catalogDAO", catalogDAO);
            }
            if (selectedScreen.equals("/newCategory")) {
               String CategoryName=req.getParameter("CategoryName");
               if(CategoryName==null)
               {
                  req.setAttribute("action","newCategory");
                  selectedScreen="/modifyCategory.jsp";

               }
               else
               {
                  Category aCategory = new
Category(1,CategoryName,req.getParameter("Description"));
                  catalogDAO.addCategory(aCategory);
                  selectedScreen="/categories.jsp";
               }
            }

            else if (selectedScreen.equals("/updateCategory")) {
               String CategoryName=req.getParameter("CategoryName");
               if(CategoryName==null)
```

```
                {

session.setAttribute("CategoryID",req.getParameter("CategoryID"));
                    req.setAttribute("action","updateCategory");
                    selectedScreen="/modifyCategory.jsp";
                }
                else
                {
                    String
categoryID=(String)session.getAttribute("CategoryID");
                    Category                                                    aCategory=new
Category(Integer.parseInt(categoryID),CategoryName,req.getParameter("
Description"));
                    catalogDAO.updateCategory(aCategory);
                    selectedScreen="/categories.jsp";
                }
            }
            else if (selectedScreen.equals("/deleteCategory"))
            {
                int                                                         categoryID
=Integer.parseInt(req.getParameter("CategoryID"));
                catalogDAO.deleteCategory(categoryID);
                selectedScreen="/categories.jsp";
            }


            RequestDispatcher                                   rd                  =
getServletContext().getRequestDispatcher(selectedScreen);
            rd.forward(req, res);

        }

          public          void     doPost(HttpServletRequest          req,     HttpServletResponse
res)
            throws ServletException, IOException {
            doGet(req, res);

          }


        }


<%@ page    contentType      ="text/html; charset=gb2312"              language     ="java"
```

```
import ="java.util.*" errorPage ="errorpage.jsp" %>
<%@ page import ="category.*" %>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title> 产品类别 </title>
</head>

<body bgcolor="#0099FF" text="#FFFFFF" link="#33FF00">

<p>   </p>
<p align="center"><font color="#00FF00" size="+3" face="华文行楷 "> 所有产品类别 </font></p>
<p><a href="newCategory"><font size="+1" face="华文行楷 "> 新加产品类别 </font></a></p>

  <div align="center">
    <table width="75%" border="1">
      <tr>
        <td> 类别编号 </td>
        <td> 类别名称 </td>
        <td> 类别描述 </td>
        <td> 删除 </td>
        <td> 更新 </td>
      </tr>
      <%
  String CategoryID= "" ;
  String CategoryName= "" ;
 CatalogDAO aCatalogDAO= new CatalogDAO();
 ArrayList al =aCatalogDAO.getCategories();
    for ( int i=0;i<al.size();i++)
 {
      Category cat =(Category)al.get(i);
    %>
        <tr>
        <td> <%=cat.getCategoryID() %></td>
        <td> <%=cat.getCategoryName() %></td>
        <td> <%=cat.getDescription() %></td>
        <td> <a href="deleteCategory?&CategoryID= <%=cat.getCategoryID() %>"> 删除 </a></td>
        <td> <a href="updateCategory?CategoryID= <%=cat.getCategoryID() %>"> 更新 </a> </td>
      </tr>
```

```
        <%
}
  %>
      </table>
  </div>

</body>
</html>
```

```
<%@ page  contentType  ="text/html; charset=gb2312"  language  ="java"
import  ="java.sql.*"  errorPage  =""  %>
<%
 String action=(String)  request  .getAttribute(  "action"  );
%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>  <%=action  %></title>
</head>
<body bgcolor="#0099FF" text="#FFFFFF" link="#33FF00">
```

```
<p align="center"><font color="#00FF00" size="+3" face="  方正舒体
"> <%=action  %></font></p>
<p align="center">     </p>
<form  method="post" action="  <%=action  %>">
  <table width="49%" height="50"  border="1" align="center"
cellpadding="0" cellspacing="0">
    <tr>
      <td width="48%">  类别名称  </td>
      <td width="52%"><input name="CategoryName" type="text" ></td>
    </tr>
    <tr>
      <td width="48%">  类别描述  </td>
      <td width="52%"><input name="Description" type="text" ></td>
    </tr>
  </table>
  <p align="center">
    <input type="submit" name="Submit" value="  提交 ">
  </p>
</form>
<p>    </p>
```

```
<p><a href="getCategories.jsp">                    &lt;&lt;Back        </a></p>
  </body>
</html>
```

## 代码分析与讨论

**1**）创建 **PreparedStatement**。PreparedStatement与 Statement 类似，都是使用 Connection 对象创建的。不同处在于，对于 PreparedStatement，必须在创建时而不是执行时指定 SQL 语句。以上 updateCategor方法中的加底纹部分的代码显示了如何创建一个 PreparedStatement 对象。该例子中，“？”用做变量的占位符， 在执行 PreparedStatement之前，需要使用 setXXX 方法用值替代所有的占位符。

**2**）使用 **PreparedStatement** 执行更新。 创建了 PreparedStatement 对象后，就可以执行它了。与 Statement 类似，可以通过调用 PreparedStatement对象的 executeUpdate()方法执行更新或 executeQuery()执行查询。不像执行 Statement 的情形，不需要向 executeUpdate 或 executeQuery 方法传递任何东西。 如下的代码是 updateCategor (Category aCategory)方法中的代码段，它在执行 PreparedStatement 之前调用 setXXX 方法为每个占位符设置值（只有为 PreparedStatement每个占位符设置了值之后，才能执行 PreparedStatement）。

```
pstmt.setString(1,aCategory.getCategoryName());
pstmt.setString(2,aCategory.getDescription());
pstmt.setInt(3,        aCategory.getCategoryID());
pstmt.executeUpdate();
```

注意：

（1）在执行 PreparedStatement之前，必须为在创建 PreparedStatement时定义的占位符提供实际的数据。可以调用 PreparedStatement 对象中相应的 setXXX 方法完成这项任务。setXXX 方法的第一个参数指定要替换的占位符的索引（索引从 1 开始），pstmt.setString(1,aCategory.getCategoryName()) 将第一个占位符（索引为 1）换为 aCategory.getCategoryName()的值， pstmt.setString(2, aCategory.getCategoryName()) 将第二个占位符（索引为 2）换为 aCategory.getCategoryName() 的值， pstmt.setInt(3, aCategory.getCategoryID() ) 将第三个占位符（索引为 3）换为 aCategory.getCategoryID() 的值。

（2）一旦将 PreparedStatement参数（占位符）设置为一给定值，只有重新设置成另一个值或者调用 clearParameters()方法，此参数值才会改变。否则，下次再执行此 PreparedStatement对象时使用上次的值。

示例 **1**：如下的代码是 deleteCategory()方法中的代码段，它是创建和执行 PreparedStatement的另一个示例。执行了 pstmt 之后， Categories 数据库表中 CategoryID 值为 categoryId 变量值的记录被删除。

```
String qstr = "Delete Categories  Where CategoryID =? ";
getDBConnection();
pstmt = dbConnection.prepareStatement(qstr);
pstmt.setInt(1,categoryId);
pstmt.executeUpdate();
```

示例 **2**：如下的代码是 addCategory(Category aCategory)方法中的代码段 ,它也是创建和

执行 PreparedStatement的另一个示例。执行了 pstmt 之后，在 Categories 数据库表中添加了一记录， 该记录 CategoryName 字段值为 aCategory.getCategoryName()的值， Description 字段值为 aCategory.getDescription 的值。

```
String qstr = "INSERT INTO Categories ";
qstr += "(CategoryName,Description)";
qstr += " VALUES(?,?)";
PreparedStatement pstmt = null;
getDBConnection();
pstmt = dbConnection.prepareStatement(qstr);
pstmt.setString(1,aCategory.getCategoryName());
pstmt.setString(2,aCategory.getDescription());
pstmt.executeUpdate();
```

3）使用 **PreparedStatement** 执行查询。 创建 PreparedStatement对象时， SQL 语句既可以是更新语句， 也可以是查询语句， 既可以是带参数的 SQL 语句，也可以是不带参数的 SQL 语句。如果创建 PreparedStatement对象时 SQL 语句是查询语句， 则必须调用 executeQuery()方法执行查询， 否则调用 executeUpdate()方法。以下代码是 getCategories()方法中的代码段，该代码段创建 PreparedStatement 对象的 SQL 语句是不带参数的查询语句，因此，执行该 PreparedStatement对象时要调用 executeQuery()方法。由于创建 PreparedStatement对象的 SQL 语句是不带参数，没有占位符，因此在执行 PreparedStatement前，不需调用 setXXX 方法。

```
getDBConnection();
String qstr =" select CategoryID, CategoryName , Description from Categories ";
PreparedStatement pstmt = dbConnection.prepareStatement(qstr);
ResultSet rs = pstmt.executeQuery();
```

以下代码是 getCategory(int categoryId) 方法中的代码段， 该代码段创建 PreparedStatement 对象的 SQL 语句是带参数的查询语句，因此，执行该 PreparedStatement 对象时要调用 executeQuery()方法。由于创建 PreparedStatement对象的 SQL 语句是带参数，有占位符，因此在执行 PreparedStatement前，需要调用 setXXX 方法。

```
String qstr = "select CategoryID, CategoryName, Description from Categories "
+ " where CategoryID =? ";
PreparedStatement pstmt = null;
ResultSet rs = null;
getDBConnection();
pstmt=dbConnection.prepareStatement(qstr);
pstmt.setInt(1,categoryId);
rs = pstmt.executeQuery();
```

3）执行 **PreparedStatement** 的返回值。 执行 PreparedStatement和执行 Statement返回值类型完全一样， PreparedStatement.executeQuery()方法返回一个包含查询结果的 ResultSet 对象，如 getCategories()方法中调用 PreparedStatement.executeQuery()方法返回一个包含查询结果的 ResultSet 对象。 PreparedStatement.executeUpdate()方法返回的值是一个整数，指出表中有多少行做了更新。 当使用 executeUpdate()方法执行 DLL 语句时， 例如， CREATE TABLE ，则返回 0

## 11.4 使用 CallableStatement执行存储过程

存储过程是数据库中执行任务的非常有效的方式。 存储过程是预编译的存储在数据库中并在数据库中执行， 所以只有很少的数据访问开销。 尽管对编写存储过程的讨论超出了本书的范围， 但是一定要记住应当尽量选择编写存储过程并访问它们， 以便执行有大量数据的操作。

JDBC 可以在 Java 应用程序中执行存储过程，向它传递值并获得结果。为此，要使用 CallableStatement。与以往一样，使用 Connection 对象创建 CallableStatement 对象，要创建一个 CallableStatement 对象，要调用 Connection 对象的 prepareCall 方法。prepareCall 方法返回一个 CallableStatement 对象，这个方法以一个 String 为参数，它表明了要用一种特殊的转义语法调用的存储过程。如果调用的存储过程不返回结果，那么调用存储过程的语法如下：

{call procedure_name(?,?, … )}

这里，问号（ ?）表示存储过程的输入输出参数， 参数是可选的， 存储过程可以没有任何输入并不返回任何输出。 procedure_name 是数据库中存储过程的实际名称。 如果存储过程返回数据，那么语法则是：

{? = call procedure_name(?,?, … )}

一个没有任何输入 /输出参数并不返回任何输出的存储过程可用以下语法调用：

{call procedure_name}

### 11.4.1 IN 参数

调用存储过程时，使用 Connection 的 prepareCall 方法创建 CallableStatement 对象对象，prepareCall 方法的创数是转义字符串，使用？作为每个输入参数（ IN 参数）的占位符。使用 CallableStatement 对象的相应 setXXX 方法为每个参数设置值， 这些 setXXX 方法以 IN 参数的实际名字和它的实际值为参数。 setXXX 方法的第一个参数还可以是 IN 参数的索引 （从 1 开始）。最后调用 CallableStatement 对象的 executeUpdate()方法或 executeQuery()方法执行 CallableStatement。

```
CREATE Procedure ProductsByCategory
(
   @CategoryID int
)
AS
SELECT *
FROM Products
WHERE CategoryID = @CategoryID
ORDER BY
   ModelName,
   ModelNumber
```

```
package category;
public class Product {
private String productid,categoryid,
                        modelnumber,modelname,
                        productimage,unitcost,
                        description;

Product(){}
public String getProID(){
        return productid;
}
public void setProID(String id){
        productid=id;
}
public String getCateID(){
        return categoryid;
}
public void setCateID(String id){
        categoryid=id;
}
public String getNumber(){
        return modelnumber;
}
public void setNumber(String num){
        modelnumber=num;
}
public String getName(){
        return modelname;
}
public void setName(String name){
        modelname=name;
}
public String getImage(){
        return productimage;
}
public void setImage(String image){
        productimage=image;
}
public String getUnitCost(){
        return unitcost;
}
```