第5章 JDBC技术

【章概述】

Web应用中最重要的数据仓库就是数据库,在JSP页面中可以通过JDBC技术连接数据库存取数据。数据的存取方式及效率对Web应用的性能有着重要的影响。

本章首先通过一个实例介绍JDBC访问数据库的步骤以及驱动程序等基本概念,然后使用JDBC技术在网络点餐系统的菜品管理模块中实现读取、增加、修改、删除菜品信息等功能。

【教学重点与难点】

重点:

(1)JDBC常用接口及类的使用。

(2)JDBC访问数据库的步骤。

(3)实现网络点餐系统菜品管理模块的基本操作。

难点:

JDBC常用接口及类的使用。

【知识单元正文】

5.1 认识JDBC

JDBC是一套面向对象的应用程序接口,它制定了统一的访问各类关系数据库的标准接口,为各个数据库厂商提供标准接口的实现。通过使用JDBC技术,开发人员可以用纯Java语言和标准的SQL语句编写完整的数据库应用程序,并且真正实现了软件的跨平台性。在Web应用开发过程中,程序员可以使用JDBC中的类与接口来连接多种关系型数据库,进行数据库操作,避免了使用不同的数据库时,需要重新编写连接数据库程序的麻烦。

5.1.1 项目构思

编写网页jdbc.jsp,使用JDBC技术连接MySQL数据库mealsystem,实现对user表中信息查询和显示功能,查询结果显示在表格中。

5.1.2　项目设计

1.数据库设计

安装配置数据库、创建数据库、创建表格以及添加数据请见附录 D 及见表 1-1 至 1-4。

2.程序设计

在 MyEclipse 下新建 Web Project ch05，将 MySQL 的 JDBC 驱动程序复制到 ch05 的 WebRoot\WEB-INF\lib 目录下。MySQL 的 JDBC 驱动程序 mysql-connector-java.jar 在本教材的配套资源的开发工具目录下。

在 jdbc.jsp 页面导入 java.sql 这个包，然后通过 Class.forName()加载数据库驱动，通过 DriverManager.getConnection()建立与数据库的连接。通过得到的连接对象创建语句对象 Statement，执行查询 user 表中所有信息的 SQL 语句，将查询结果 ResultSet 对象中的 user 表的内容显示在表格中。

5.1.3　项目实施

在 ch05\WebRoot 目录下新建文件夹 mealsystem，在 mealsystem 下新建文件 jdbc.jsp，代码如下：

文件名：jdbc.jsp

```
<%@ page contentType="text/html;charset=gb2312" language="java"  import="java.util.*,
java.sql.*,java.io.*"%>
<html>
<head>
<meta http-equiv="Content-Language" content="zh-cn">
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>用户表</title>
</head>
<body>
<center>
<p>用户表</p>
<table border="1" width="76%" id="table1" bordercolor="green" align="center">
<tr>
<td>id 号</td>
<td>姓名</td>
<td>密码</td>
<td>角色</td>
<td>电话</td>
<td>地址</td>
</tr>
<%
Class.forName("com.mysql.jdbc.Driver");//(1)装载驱动程序
Connection con=DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/mealsystem",
```

```
"root","root");//(2)创建连接,mealsystem 是数据库名称,两个 root 分别为用户名和密码。
Statement stmt=con.createStatement();
ResultSet rs=stmt.executeQuery("select * from user");
//(3)发送 SQL 语句到数据库中,user 是数据库中表名。
while(rs.next())
{
int id=rs.getInt("id");//(4)处理数据并查询结果。
String userName=rs.getString("userName");
String password =rs.getString("password");
int ident=rs.getInt("ident");
String telephone=rs.getString("telephone");
String address=rs.getString("address");
%>
<tr>
<td><%=id%></td>
<td><%=userName%></td>
<td><%=password%></td>
<td><% if(ident==1){
        out.print("管理员");
        }
        else if(ident==0){
        out.print("普通用户");
        }
    %></td>
<td><%=telephone%></td>
<td><%=address%></td>
</tr>
<%
}
rs.close(); //(5)关闭
stmt.close();
con.close();
%>
</table>
</center>
</body>
</html>
```


5.1.4 项目运行

在浏览器地址栏中输入 URL:http://127.0.0.1:8080/ch05/mealsystem/jdbc.jsp,得到文件 jdbc.jsp 的运行结果如图 5-1 所示。

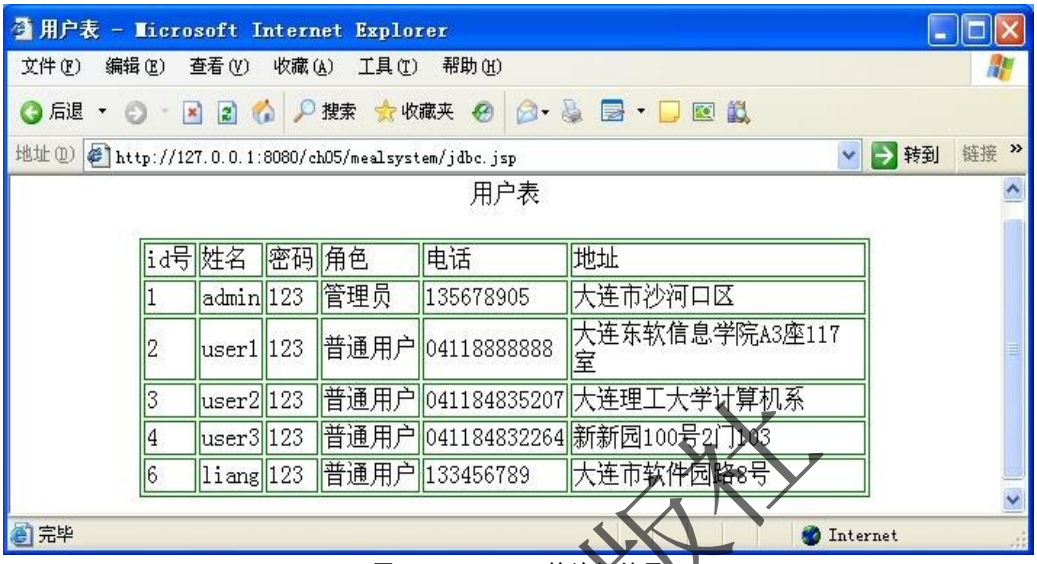

图 5-1　jdbc.jsp 的访问结果

5.1.5　知识点

1. JDBC 概述

JDBC(Java DataBase Connectivity)称为 Java 数据库连接，它是一种用于数据库访问的应用程序 API，由一组用 Java 语言编写的类和接口组成。有了 JDBC 就可以用统一的语法对多种关系数据库进行访问，而不用担心其数据库操作语言的差异。换言之，有了 JDBC，就不必为访问 MySQL 数据库专门写一个程序，为访问 Oracle 又专门写一个程序等等，只需用 JDBC 写一个程序就够了。

2. JDBC 任务

简单地说，JDBC 能完成下列三件事：

(1)同一个数据库建立连接；

(2)向数据库发送 SQL 语句；

(3)处理数据库返回的结果。

3. JDBC 数据库驱动程序

数据库厂商一般会提供一组 API 访问数据库。流行数据库如 Oracle、SQL Server、Sybase 和 Informix 都为客户访问提供了专用的 API。有些厂商也专门提供数据库驱动程序，并且这些产品除了执行驱动的功能外，往往还提供额外的服务。

有 4 种类型的数据库驱动程序，它们分别是：

(1)类型 1：JDBC-ODBC 桥

JDBC-ODBC 桥是 SUN 公司提供的，是 JDK 提供的标准 API。这种类型的驱动实际是把所有 JDBC 的调用传递给 ODBC，再由 ODBC 调用本地数据库驱动代码。只要本地机装有相关的 ODBC 驱动，那么采用 JDBC-ODBC 桥可以访问所有的数据库。但是，由于 JDBC-ODBC 先调用 ODBC，再由 ODBC 去调用本地数据库接口访问数据库，所以执行效率比较低，对于那些大数据量存取的应用是不适合的。而且，这种方法要求客户端必须安装 ODBC

驱动,所以对于基于 Internet 的应用也是不合适的。因为,不可能要求所有客户都能找到 ODBC 驱动。JDBC-ODBC 桥如图 5-2 所示。

图 5-2 JDBC 类型 1 驱动程序连接

(2)类型 2:本地 API 驱动

本地 API 驱动直接把 JDBC 调用转变为数据库的标准调用再去访问数据库。这种方法需要本地数据库驱动代码。本地 API 驱动如图 5-3 所示。

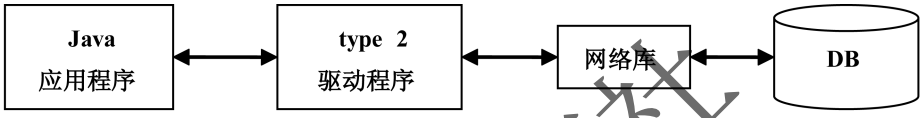

图 5-3 JDBC 类型 2 驱动程序连接

这种驱动比起 JDBC-ODBC 桥执行效率大大提高了。但是,它仍然需要在客户端加载数据库厂商提供的代码库,这样就不适合基于 Internet 的应用。并且它的执行效率比起 3、4 型的 JDBC 驱动还是不够高。

(3)类型 3:网络协议驱动

这种驱动实际上是根据三层结构建立的。JDBC 先把对数据库的访问请求传递给网络上的中间件服务器。中间件服务器再把请求翻译为符合数据库规范的调用,再把这种调用传给数据库服务器。如果中间件服务器也是用 Java 开发的,那么在中间层也可以使用 1、2 型 JDBC 驱动程序作为访问数据库的方法。网络协议驱动—中间件服务器—数据库服务器,如图 5-4 所示。

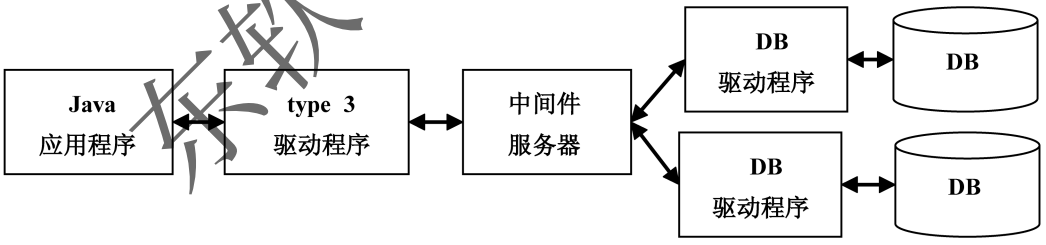

图 5-4 JDBC 类型 3 驱动程序连接

由于这种驱动是基于服务器的,所以它不需要在客户端加载数据库厂商提供的代码库,而且它在执行效率和可升级性方面是比较好的。因为大部分功能实现都在服务器端,所以这种驱动可以设计的很小,可以非常快速的加载到内存中。但是,这种驱动在中间件层仍然需要配置其他数据库驱动程序,并且由于多了一个中间层传递数据,它的执行效率还不是最好。

(4)类型 4:本地协议驱动

这种驱动直接把 JDBC 调用转换为符合相关数据库系统规范的请求。由于 4 型驱动写的应用可以直接和数据库服务器通讯,这种类型的驱动完全由 Java 实现,因此实现了平台独立性。本地协议驱动—数据库服务器,如图 5-5 所示。

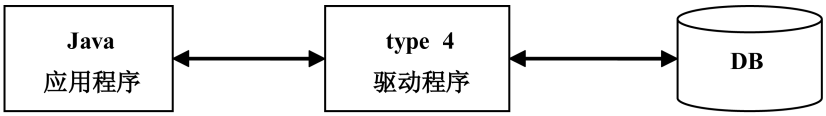

图 5-5　JDBC 类型 4 驱动程序连接

由于这种驱动不需要先把 JDBC 的调用传给 ODBC 或本地数据库接口或者是中间层服务器。所以它的执行效率是非常高的。而且，它根本不需要在客户端或服务器端装载任何的软件或驱动。这种驱动程序可以动态的被下载，但是对于不同的数据库需要下载不同的驱动程序。

以上对四种类型的 JDBC 驱动进行了说明。那么它们适合哪种类型的应用开发呢？JDBC-ODBC 桥由于它的执行效率不高更适合做为开发应用时的一种过度方案，或着对于初学者了解 JDBC 编程也较适用。对于那些需要大数据量操作的应用程序则应该考虑 2、3、4 型驱动。在 Intranet 方面的应用可以考虑 2 型驱动，但是由于 3、4 型驱动在执行效率上比 2 型驱动有着明显的优势，而且目前开发的趋势是使用纯 Java，所以 3、4 型驱动也可以作为考虑对象。至于基于 Internet 方面的应用就只有考虑 3、4 型驱动了。因为 3 型驱动可以把多种数据库驱动都配置在中间层服务器，所以 3 型驱动最适合那种需要同时连接多个不同种类的数据库，并且对并发连接要求高的应用。4 型驱动则适合那些连接单一数据库的工作组应用。

4. JDBC 访问数据库步骤

JDBC 是一种规范，遵循 JDBC 接口规范，各个数据库厂家各自实现自己的驱动程序。应用在获取数据库连接时，需要以 URL 的方式制定是哪种类型的驱动，或按照固定的接口操作不同类型的数据库，JDBC 提供了一组类和接口用于对数据库的访问，用 JDBC 访问数据库需要如下几个步骤：

(1)装载驱动程序；

(2)定义连接数据库的地址；

(3)建立与数据库的连接；

(4)建立语句对象；

(5)声明并执行 SQL 语句；

(6)处理返回的结果；

(7)关闭连接并处理异常。

下面对每一步骤做如下解析：

(1)装载驱动程序

任何一种数据库驱动程序都提供一个 java.sql.Driver 接口的驱动类，在加载某个数据库驱动程序的驱动类时，都创建自己的实例对象并向 java.sql.DriverManager 类注册该实例对象。可以利用 Class.forName()方法加载某一个数据库的驱动程序。例如：

```
Class.forName("com.mysql.jdbc.Driver ");   //Mysql JDBC Driver
```

(2)定义连接数据库的地址

通过此步骤，定义要连接的数据库资源。例如，MySQL 数据的连接地址如下：

```
String MySQLURL= "jdbc:mysql://host:port/dbName";
```

连接地址由三部分组成，格式如下：jdbc:＜子协议＞:＜子名称＞。

jdbc:JDBC中的协议就是jdbc。

<子协议>:数据库驱动程序名或数据库连接机制的名称。

<子名称>:一种标记数据库的方法。子名称根据子协议的不同而不同,使用子名称的目的是定位数据库。

注意:不同数据库连接地址的命名方式是不同的。

(3)建立与数据库的连接

DriverManager类用来装载驱动程序,它所有的成员都是静态成员,所以在程序中无须对它进行实例化,直接通过类名就可以访问。DriverManager是JDBC的管理层,作用于用户和驱动程序间加载驱动程序,DriverManager类跟踪可用的驱动程序,并在数据库和相应的驱动程序之间建立连接。

DriverManager调用getConnection()方法来建立于数据库的连接,当发出连接请求时,DriverManager将检查每一个驱动程序,使用如下语句建立连接:

```
Connection con=DriverManager.getConnnection(url,username,password);
```

例如:连接MySQL的mealsystem数据库,用户名为root,密码为root。

```
String driverStr="com.mysql.jdbc.Driver";
String connStr="jdbc:mysql://127.0.0.1:3306/mealsystem";
Class.forName(driverStr);
Connection conn = DriverManager.getConnection(connStr, "root", "root");
```

(4)建立语句对象

java.sql.Statement接口用来执行静态的SQL语句,并返回执行结果。

对于insert、update和delete语句,调用executeUpdate(String sql)方法,它返回的是所影响的纪录的个数;而select语句则调用executeQuery(String sql)方法,并返回一个永远不能为null的ResultSet实例。

execute()方法也可以执行SQL语句,它返回的是一个是否有结果集的布尔值。execute()方法常用于动态的处理未知的SQL语句,事先无法知道该SQL语句的具体类型及执行的返回值。既可以执行查询语句,也可以执行更新语句。当SQL语句的执行结果是一个ResultSet结果集时,本方法返回true;并可以通过Statement的getResultSet()方法得到返回的结果集;当SQL语句执行后没有返回的结果集时,该方法返回false。

Connection接口提供了生成Statement实例的方法,一般情况下通过connection.createStatement()方法就可以得到Statement的实例。例如:

```
//创建语句对象
statement =conn.createStatement();
//构造SQL语句字符串
Strings="insert into food(foodName)value('四川凉菜') ";
//执行SQL语句
statement.executeUpdate(s);
```

Connection还提供许多方法,其中常用的方法有:

①prepareStatement()。

java.sql.PreparedStatement接口继承并扩展了Statement接口,用来执行动态的SQL

语句，即包含参数的 SQL 语句。通过 PreparedStatement 实例执行的动态 SQL 语句，将被预编译并保存到 PreparedStatement 实例中，从而可以反复并且高效地执行该 SQL 语句。例如向 mealsystem 数据库的 user 表中插入数据：

```
PreparedStatement pstm=conn.prepareStatement("insert into user values(null,?,?,?,?,?)");
pstm.setString(1,"zhangsan");
pstm.setString(2,"123");
pstm.setString(3,"0");
pstm.setString(4,"18923456543");
pstm.setString(5,"大连东软信息学院");
```

需要注意的是，在通过 setXXX()方法为 SQL 语句中的参数赋值时，建议利用与参数类型匹配的方法，也可以利用 setObject()方法为各种类型的参数赋值。setXXX()方法的第一个参数为欲赋值参数的索引位置，从 1 开始；第二个参数为参数的值。

PreparedStatement 实例的 executeQuery()方法用于执行包含参数的动态 select 语句，并返回一个永远不能为 null 的 ResultSet 实例；executeUpdate()方法用于执行包含参数的动态 insert、update 或 delete 语句，并返回一个 int 型数值，为同步更新记录的条数。

②setAutoCommit()

设置 Connection 的自动提交模式。默认为 true，即自动将更改同步到数据库中。如果设为 false，需要通过执行 commit()或 rollback()方法手动将更改同步到数据库中。

③commit()

将从上一次提交或回滚之后进行的所有更改同步到数据库，并释放 Connection 实例当前拥有的所有数据库锁定。

④rollback()

取消当前事务中的所有更改，并释放当前 Connection 实例拥有的所有数据库锁定。该方法只能在非自动提交模式下使用，如果在自动提交模式下执行该方法，将抛出异常。

(5)声明并执行 SQL 语句

SQL 语句可以是静态的，也可以是动态的。静态 SQL 语句的执行可以借助语句对象 Statement，动态 SQL 语句的执行可以借助 PreparedStatement。

(6)对结果集进行处理

java.sql.ResultSet 接口类似于一个数据表，通过该接口的实例可以获得查询结果集。ResultSet 实例通过执行查询 SQL 语句生成。

ResultSet 实例具有指向当前数据行的指针，最初，指针指向第一行记录的前方，通过 next()方法可以将指针移动到下一行，如果存在下一行该方法则返回 true，否则返回 false，所以可以通过 while 循环来迭代 ResultSet 结果集。默认情况下 ResultSet 实例不可以更新，只能向前移动指针，所以只能迭代一次，并且只能按从前到后的顺序。如果需要，可以生成可滚动和可更新的 ResultSet 实例，这样就可以通过 first()、last()、previous()、next()和 absolute(int i)等方法浏览可滚动结果集中的内容，也可通过 updateString(int col,String val)等方法对结果集中的内容进行修改。

ResultSet的几个主要方法：

①next()

用于将ResultSet定位到下一行。ResultSet最初是定位在第0行上的，此时应该先调用next()方法将其定位到第1行上，然后才可输出。当ResultSet已经到了最后一行时，再调用next()方法，则返回false。

②getXxx()

当ResultSet已经定位在某一行上时，使用getXxx()方法得到这一行上单个字段的值。针对不同的字段类型，调用不同的getXxx(String s)或getXxx(int i)方法。

例如，如果读取表中记录的第2个字段，字段名为“userName”，是文本型的，则可以使用getString(2)或getString("userName")来得到它的值。

(7)关闭连接并处理异常

在操作完数据库后，都要及时关闭数据库连接，释放占有的数据库和JDBC资源，以免影响软件的运行速度。ResultSet、Statement和Connection接口均提供了关闭各自实例的close()方法，ResultSet、Statement和Connection实例的关闭顺序如图5-6所示。

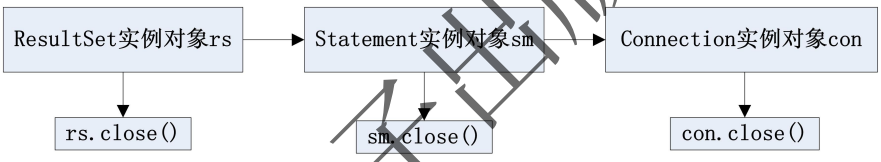

图5-6 ResultSet、Statement和Connection实例的关闭次序

虽然Java的垃圾回收机制会定时清理缓存、关闭长时间不用的数据库连接，但是如果不及时关闭，数据库连接达到一定数量，将严重影响数据库和计算机运行速度，甚至瘫痪。

注意：关闭对象前需要判断所关闭的对象是否为空。

5.JDBC的优点和缺点

JDBC API用于连接Java应用程序与各种关系数据库。这使得人们在建立客户/服务器应用程序时，通常把Java作为编程语言，把任何一种浏览器作为应用程序的友好界面，把Internet或Intranet作为网络主干，把有关的数据库作为数据库后端。

使用JDBC具有如下优点：

(1)JDBC API与ODBC十分相似，有利于用户理解。

(2)JDBC使得编程人员从复杂的驱动器调用命令和函数中解脱出来，可以致力于应用程序中的关键地方。

(3)JDBC支持不同的关系数据库，使得程序的可移植性大大加强。

(4)用户可以使用JDBC-ODBC桥驱动器将JDBC函数调用转换为ODBC。

(5)JDBC API是面向对象的，可以让用户把常用的方法封装为一个类以备后用。

使用JDBC具有如下缺点：

(1)使用JDBC，访问数据记录的速度会受到一定程度的影响。

(2)JDBC结构中包含了不同厂家的产品，这就给更改数据源带来了很大的麻烦。

5.2　使用 JDBC 实现网络点餐系统的菜品管理

5.2.1　项目构思

使用 JDBC 技术连接 MySQL 数据库 mealsystem，实现网络点餐系统的菜品管理模块，完成菜品信息的读取、增加、修改、删除功能。

5.2.2　项目设计

为实现项目功能，共需要设计 7 个 JSP 文件，对应的文件名和功能描述如表 5-1 所示。

表 5-1　　项目的文件说明表

文件名	描述
head.jsp	提供网络点餐系统各个管理模块的超级链接
manageFood.jsp	显示所有菜品信息页面，可以按照菜品分类查找，并提供添加、修改、删除菜品信息的链接
addFood.jsp	添加菜品信息页面
addFood_do.jsp	添加菜品信息的处理页面
editFood.jsp	修改菜品信息的表单页面，该页面会显示预修改的菜品信息
editFood_do.jsp	修改菜品信息的处理页面
deleteFood.jsp	删除菜品信息的页面

每个页面的具体设计如下：

(1)head.jsp 页面提供网络点餐系统的用户管理、菜品管理、菜品分类管理、查看用户点餐情况和退出点餐系统的超级链接。

(2) manageFood.jsp 页面是显示菜品信息的页面，利用 JDBC 技术连接数据库 mealsystem，读取 foodType 表中的菜品分类信息显示在下拉列表中，并提供按照分类查询菜品信息功能。利用 JDBC 技术读取 food 表中的菜品信息显示在表格中，同时提供添加、修改、删除菜品功能的超级链接。

(3)addFood.jsp 页面是添加菜品信息的表单页面，用户点击添加按钮后，通过 action 属性把请求提交到 addFood_do.jsp 页面。

(4)addFood_do.jsp 页面是添加菜品信息的处理页面，使用 request.getParameter()方法获取 addFood.jsp 页面提交的信息，使用 JDBC 执行连接数据库的系列操作，并创建实现添加功能的 SQL 语句，使用语句对象的 executeUpdate()方法执行数据的插入操作。添加成功后利用 response.sendRedirect()方法跳转到 manageFood.jsp 页面。否则利用 response.setHeader("Refresh","5;url=manageFood.jsp"),5 秒后页面自动刷新并跳转到 manageFood.jsp 页面。

(5)editFood.jsp 页面使用 request.getParameter()方法读取 manageFood.jsp 页面传

递的菜品 id 的参数值,然后使用 JDBC 连接到数据库,查找到对应的菜品信息,并使用结果集的 getString()方法从数据库中获取字段值显示在 form 表单中。用户输入修改信息后,点击修改按钮把请求提交到 editFood_do.jsp。

(6)editFood_do.jsp 页面是修改菜品信息的处理页面,使用 request.getParameter()方法获取 editFood.jsp 页面提交的信息,使用 JDBC 执行连接数据库的系列操作,并创建实现修改功能的 SQL 语句,使用语句对象的 executeUpdate()方法执行数据的更新操作。修改成功后利用 response.sendRedirect()方法跳转到 manageFood.jsp 页面。否则利用 response.setHeader("Refresh","5;url=manageFood.jsp"),5 秒后页面自动刷新并跳转到 manageFood.jsp 页面。

(7)deleteFood.jsp 页面使用 request.getParameter()方法读取 manageFood.jsp 页面传递的菜品 id 的参数值,使用 JDBC 执行连接数据库的系列操作,并创建实现删除功能的 SQL 语句,使用语句对象的 executeUpdate()方法执行数据的删除操作。删除成功后利用 response.sendRedirect()方法跳转到 mangeFood.jsp 页面。否则利用 response.setHeader("Refresh","5;url=manageFood.jsp")语句,5 秒后页面自动刷新并跳转到 manageFood.jsp 页面。

5.2.3 项目实施

在 ch05\WebRoot\mealsystem 目录下新建文件夹 admin,在 admin 目录下新建文件 head.jsp,代码如下:

文件名:head.jsp

```
<%@ page pageEncoding="GBK"%>
<table width="80%">
<tr>
<th><a href="manageUser.jsp">用户管理</a></th>
<th><a href="manageFoodType.jsp">菜品分类管理</a></th>
<th><a href="manageFood.jsp">菜品管理</a></th>
<th><a href="showDiningCar.jsp">查看用户点餐情况</a></th>
<th><a href="../logout.jsp">退出系统</a></th>
</tr>
</table>
```

在 ch05\WebRoot\mealsystem\admin 目录下新建文件 manageFood.jsp,代码如下:

文件名:manageFood.jsp

```
<%@ page pageEncoding="GBK"%>
<%@ page import="java.sql.*"%>
<html>
<head><title>菜品管理页面</title></head>
<body>
<center>
<h1><font color="red">菜品管理</font></h1>
<hr color="green"/>
```

```
<%@ include file="head.jsp" %>
<p>
<form action="manageFood.jsp" method="post">
<a href="addFood.jsp">添加菜品</a>  
<select name="foodType">
<%
    Class.forName("com.mysql.jdbc.Driver");
    String url="jdbc:mysql://localhost:3306/mealsystem";
    String dbun="root";String dbpw="root";
    Connection con=DriverManager.getConnection(url,dbun,dbpw);
    String sql="select * from foodtype";
    PreparedStatement pstmt=con.prepareStatement(sql);
    ResultSet rs=pstmt.executeQuery();
%>
    <option value=""></option>
<%
    while(rs.next()){
       out.print("<option value=");
       int id=rs.getInt(1);
       out.print(id);
       String foodType=request.getParameter("foodType");
       if(foodType!=null
       && !foodType.equals("")
       && Integer.parseInt(foodType)==id){
           out.print(" selected>");
       }else{
          out.print(">");
       }
       out.print(rs.getString(2));
       out.println("</option>");
    }
    rs.close();pstmt.close();
%>
</select>
<input type="submit" value="按分类查询"/>
</form>
<p>
<table border width="80%">
<tr>
<th>序号</th><th>菜品名称</th><th>价格</th>
<th>类型</th><th>点餐率:次</th><th>备注</th>
<th>操作</th>
```

```
</tr>
<%
    String foodType=
    request.getParameter("foodType");
    if(foodType==null || foodType.equals("")){
        sql="select f.*,ft.typeName from food f,foodtype ft where f.type=ft.id";
        pstmt=con.prepareStatement(sql);
    }else{
        sql="select f.*,ft.typeName from food f,foodtype ft where f.type=ft.id and ft.id=?";
        pstmt=con.prepareStatement(sql);
        pstmt.setString(1,foodType);
    }
    rs=pstmt.executeQuery();
    int i=1;
    while(rs.next()){
        int id=rs.getInt("id");
        String foodName=rs.getString("foodName");
        int price=rs.getInt("price");
        int hits=rs.getInt("hits");
        int comment=rs.getInt("comment");
        String typeName=rs.getString("typeName");
%>
        <tr>
        <td><%=i%></td>
        <td><%=foodName%></td>
        <td><%=price%></td>
        <td><%=typeName%></td>
        <td><%=hits%></td>
        <td>
        <%
            if(comment==-1){
                out.print(" ");
            }else if(comment==0){
                out.print("厨师推荐");
            }else if(comment>0){
                out.print("特价"+comment+"元");
            }
        %>
        </td>
        <td>
        <a href="editFood.jsp?foodId=<%=id%>">修改</a>
        <a href="deleteFood.jsp?foodId=<%=id%>"
```

```
            onClick='return confirm("确定要删除吗?")'>
            删除</a>
            </td>
            </tr>
    <%
            i++;
        }
        rs.close();pstmt.close();con.close();
    %>
    </table>
    </center>
    </body>
    </html>
```

在 ch05\WebRoot\mealsystem\admin 目录下新建文件 addFood.jsp 和 addFood_do.jsp，代码如下：

文件名：addFood.jsp

```
<%@ page pageEncoding="GBK" %>
<%@ page import="java.sql.*" %>
<html>
<head><title>菜品管理——添加菜品</title></head>
<body>
<center>
<h1><font color="red">菜品管理——添加菜品</font></h1>
<hr color="green"/>
<%@ include file="head.jsp" %>
<p>
<form action="addFood_do.jsp" method="post">
<table border width="50%">
<tr>
<td>菜名</td>
<td><input type="text" name="foodName"/></td>
</tr>
<tr>
<td>特色</td>
<td><textarea name="feature" rows="4" cols="20"></textarea></td>
</tr>
<tr>
<td>食材</td>
<td><textarea name="material" rows="4" cols="20"></textarea></td>
</tr>
<tr>
<td>价格</td>
```

```
<td><input type="text" name="price"/></td>
</tr>
<tr>
<td>类型</td>
<td><select name="type">
<%
    Class.forName("com.mysql.jdbc.Driver");
    String url="jdbc:mysql://localhost:3306/mealsystem";
    String dbun="root";String dbpw="root";
    Connection con=DriverManager.
    getConnection(url,dbun,dbpw);
    String sql="select * from foodtype";
    PreparedStatement pstmt=
    con.prepareStatement(sql);
    ResultSet rs=pstmt.executeQuery();
    while(rs.next()){
       out.print("<option value='"+rs.getInt(1)+"'>");
       out.print(rs.getString(2));
       out.println("</option>");
    }
    rs.close();pstmt.close();con.close();
%>
</select></td>
</tr>
<tr><td>图片</td>
<td><input type="text" name="picture"/></td>
</tr>
<tr><td>备注</td>
<td>
<input type="text" name="comment" value="-1">
<br/>
<font color="red" size="2">
-1表示正常菜,0表示厨师推荐,正整数为特价菜价格。
</font>
</td>
</tr>
<tr>
<td align="center" colspan="2">
<input type="submit" value="添加"/>
</td>
</tr>
</table>
```

```
</form>
</center>
</body>
</html>
```

文件名:addFood_do.jsp

```
<%@ page pageEncoding="GBK" %>
<%@ page import="java.sql.*" %>
<%
    String url="jdbc:mysql://localhost:3306/mealsystem";
    String dbun="root";String dbpw="root";
    Connection con=DriverManager.
    getConnection(url,dbun,dbpw);
    String sql="insert into food values(null,?,?,?,?,?,?,0,?)";
    PreparedStatement pstmt=con.prepareStatement(sql);
    request.setCharacterEncoding("GBK");
    pstmt.setString(1,request.getParameter("foodName"));
    pstmt.setString(2,request.getParameter("feature"));
    pstmt.setString(3,request.getParameter("material"));
    pstmt.setString(4,request.getParameter("price"));
    pstmt.setString(5,request.getParameter("type"));
    pstmt.setString(6,request.getParameter("picture"));
    pstmt.setString(7,request.getParameter("comment"));
    int r=pstmt.executeUpdate();
    pstmt.close();con.close();
    if(r==1){
        response.sendRedirect("manageFood.jsp");
    }else{
        out.println("添加失败!5 秒后返回.....");
        response.setHeader("Refresh","5;url=manageFood.jsp");
    }

%>
```

在 ch05\WebRoot\mealsystem\admin 目录下新建文件 editFood.jsp 和 editFood_do.jsp,代码如下:

文件名:editFood.jsp

```
<%@ page pageEncoding="GBK" %>
<%@ page import="java.sql.*" %>
<html>
<head><title>菜品管理——修改菜品</title></head>
<body>
<center>
<h2><font color="red">菜品管理 &mdash;&mdash;修改菜品</font></h2>
```

```
<hr color="green"/>
<%@ include file="head.jsp" %>
<p>
<%
    String id=request.getParameter("foodId");
    String url="jdbc:mysql://localhost:3306/mealsystem";
    String dbun="root";String dbpw="root";
    Connection con=DriverManager.
    getConnection(url,dbun,dbpw);
    String sql="select * from food where id=?";
    PreparedStatement pstmt=
    con.prepareStatement(sql);
    pstmt.setString(1,id);
    ResultSet rs=pstmt.executeQuery();
    rs.next();
    String foodName=rs.getString("foodName");
    String feature=rs.getString("feature");
    String material=rs.getString("material");
    String price=rs.getString("price");
    String type=rs.getString("type");
    String picture=rs.getString("picture");
    String comment=rs.getString("comment");
    rs.close();pstmt.close();
%>
<form action="editFood_do.jsp" method="post">
<input type="hidden" name="id" value="<%=id %>"/>
<table border width="50%">
<tr>
<td>菜名</td>
<td><input type="text" name="foodName" value="<%=foodName %>"/></td>
</tr>
<tr>
<td>特色</td>
<td><textarea name="feature" rows="4" cols="20">
<%=feature %>
</textarea></td>
</tr>
<tr>
<td>食材</td>
<td><textarea name="material" rows="4" cols="20">
<%=material %>
</textarea></td>
</tr>
<tr>
```

```
<td>价格</td>
<td><input type="text" name="price" value="<%=price%>"/></td>
</tr>
<tr>
<td>类型</td>
<td><select name="type">
<%
    sql="select * from foodtype";
    pstmt=con.prepareStatement(sql);
    rs=pstmt.executeQuery();
    while(rs.next()){
       out.print("<option value='");
       int foodId=rs.getInt(1);
       out.print(foodId);
       if(Integer.parseInt(type)==foodId){
          out.print("' selected>");
       }else{
          out.print("'>");
       }
       out.print(rs.getString(2));
       out.println("</option>");
    }
    rs.close();pstmt.close();con.close();
%>
</select></td>
</tr>
<tr><td>图片</td>
<td><img src="../../<%=picture %>"/><br/>
<input type="text" name="picture" value="<%=picture %>"/></td>
</tr>
<tr><td>备注</td>
<td>
<input type="text" name="comment" value="<%=comment %>">
<br/>
<font color="red" size="2">
-1 表示正常菜,0 表示厨师推荐,正整数为特价菜价格。
</font>
</td>
</tr>
<tr>
<td align="center" colspan="2">
<input type="submit" value="修改"/>
</td>
</tr>
```

```
</table>
</form>
</center>
</body>
</html>
```

文件名:editFood_do.jsp

```
<%@ page pageEncoding="GBK" %>
<%@ page import="java.sql.*" %>
<%
    String url="jdbc:mysql://localhost:3306/mealsystem";
    String dbun="root";String dbpw="root";
    Connection con=DriverManager.
    getConnection(url,dbun,dbpw);
    String sql="update food set "+
        "foodName=?,feature=?,material=?,price=?,type=?,picture=?,comment=? where id=?";
    PreparedStatement pstmt=
    con.prepareStatement(sql);
    request.setCharacterEncoding("GBK");
    pstmt.setString(1,request.getParameter("foodName"));
    pstmt.setString(2,request.getParameter("feature"));
    pstmt.setString(3,request.getParameter("material"));
    pstmt.setString(4,request.getParameter("price"));
    pstmt.setString(5,request.getParameter("type"));
    pstmt.setString(6,request.getParameter("picture"));
    pstmt.setString(7,request.getParameter("comment"));
    pstmt.setString(8,request.getParameter("id"));
    int r=pstmt.executeUpdate();
    pstmt.close();con.close();
    if(r==1){
        response.sendRedirect("manageFood.jsp");
    }else{
        out.println("修改失败!5秒后返回.....");
        response.setHeader("Refresh","5;url=manageFood.jsp");
    }

%>
```

在ch05\WebRoot\mealsystem\admin目录下新建文件deleteFood.jsp,代码如下:

文件名:deleteFood.jsp

```
<%@ page pageEncoding="GBK" %>
<%@ page import="java.sql.*" %>
<%
    String url="jdbc:mysql://localhost:3306/mealsystem";
    String dbun="root";String dbpw="root";
    Connection con=DriverManager.
```

```
        getConnection(url,dbun,dbpw);
        String sql="delete from food where id=?";
        PreparedStatement pstmt=
        con.prepareStatement(sql);
        pstmt.setString(1,request.getParameter("foodId"));
        int r=pstmt.executeUpdate();
        pstmt.close();con.close();
        if(r==1){
            response.sendRedirect("manageFood.jsp");
        }else{
            out.println("删除失败！5 秒后返回.....");
            response.setHeader("Refresh","5;url=manageFood.jsp");
        }
%>
```

注意：所有菜品的图片存放在 ch05\WebRoot\mealsystem\images 目录下。

5.2.4　项目运行

在浏览器地址栏中输入：http://127.0.0.1:8080/ch05/mealsystem/admin/manageFood.jsp 打开菜品管理的主页面，如图 5-7 所示。

序号	菜品名称	价格	类型	点餐率:次	备注	操作
1	蒜菜炒鸡蛋	9	家常	3	厨师推荐	修改 删除
2	韭菜炒鸡蛋	8	家常	2		修改 删除
3	渝味辣白菜	6	家常	1		修改 删除
4	爆炒腰花	12	家常	0		修改 删除
5	韩国泡菜汤	16	家常	0	厨师推荐	修改 删除
6	清蒸桂鱼	25	家常	6		修改 删除
7	酸辣白菜	14	家常	1		修改 删除
8	醋溜白菜	14	家常	0		修改 删除
9	木须肉	8	家常	0		修改 删除
10	肉末豆腐	7	家常	1		修改 删除
11	小葱拌豆腐	22	凉菜	2	特价18元	修改 删除
12	泡椒鸡爪	12	凉菜	0		修改 删除

图 5-7　显示所有菜品信息页面

点击添加菜品链接，进入添加菜品信息页面，如图 5-8 所示。

图 5-8 添加菜品信息页面

在图 5-8 中输入菜品信息后，点击“添加”按钮，如果添加成功，则跳转到 manageFood.jsp 页面，类似图 5-7 所示，这里不再截图赘述。

在图 5-7 中点击某一条菜品的修改链接进入修改页面 editFood.jsp，如图 5-9 所示。

图 5-9 菜品信息修改页面

在图 5-9 中输入修改信息后，点击“修改”按钮，修改成功后自动跳转到类似图 5-7 所示的页面。

在图 5-7 中点击某一条菜品的删除链接将弹出确认删除的对话框，如图 5-10 所示。如果确定删除，则点击“确定”按钮，删除成功后跳转到类似图 5-7 所示的页面。

图 5-10　菜品信息确认删除对话框

【实践环节设计】

单元项目：使用 JDBC 实现网络点餐系统的菜品分类管理功能。

1.项目构思

使用 JDBC 技术连接 MySQL 数据库 mealsystem，实现网络点餐系统的菜品分类管理模块，完成菜品分类的浏览、添加、修改、删除功能。

2.项目设计

为实现项目功能，共需要编写 6 个 JSP 文件，manageFoodType.jsp 是菜品分类管理功能的首页面，也是浏览所有菜品分类的页面，addFoodType.jsp 是添加菜品分类的页面，addFoodType_do.jsp 是添加菜品分类信息的操作页面，editFoodType.jsp 是修改菜品分类的页面，editFoodType_do.jsp 是修改菜品分类信息的操作页面，deleteFoodType.jsp 是处理删除菜品分类信息功能页面。

(1)manageFoodType.jsp 显示菜品分类管理信息，它利用 JDBC 连接 mealsystem 数据库读取 foodType 表中的所有记录，显示在表格中。manageFoodType.jsp 页面中的“添加分类”超级链接的地址为“/ch05/mealsystem/admin/addFoodType.jsp”；“修改分类”超级链接的地址为“/ch05/mealsystem/admin/editFoodType.jsp”；“删除分类”超级链接的地址为“/ch05/mealsystem/admin/deleteFoodType.jsp”。

(2)addFoodType.jsp 页面提供一个 form 表单，表单中有“分类名称”文本框，点击“添加”按钮后，通过 action 属性把请求提交到 addFood_do.jsp 页面。

(3)addFoodType_do.jsp 页面首先使用 request.getParameter()方法读取菜品分类名称。然后使用 JDBC 执行连接数据库的系列操作，并创建实现添加的 SQL 语句。接着使用 executeUpdate()方法执行数据的插入操作。如果添加成功，则利用 response.sendRedirect()方法跳转到 manageFoodType.jsp 页面。如果添加失败，则显示“添加菜品分类信息失败!”信息后，5 秒后页面自动刷新到 manageFoodType.jsp 页面。

(4)editFoodType.jsp 页面首先使用 request.getParameter()方法读取菜品分类 id，然后使用 JDBC 连接数据库读取对应的菜品分类信息，并把需要修改的内容显示在 form 表单

中,form表单的action属性值为“editFoodType_do.jsp”。

(5)editFoodType _do.jsp页面首先使用request.getParameter()方法读取菜品分类id和名称,然后使用JDBC连接数据库更新菜品分类信息。如果更新成功,则利用response.sendRedirect()方法跳转到manageFoodType.jsp页面。如果更新失败,则显示“更新菜品分类信息失败!”信息后,5秒后页面自动刷新到manageFoodType.jsp页面。

(6)deleteFoodType.jsp页面首先使用request.getParameter()方法读取菜品分类id,然后使用JDBC连接数据库删除相应的菜品分类下的所有菜品,接着再删除相应的菜品分类。如果删除成功,则利用response.sendRedirect()方法跳转到manageFoodType.jsp页面;如果删除失败,则显示“删除菜品分类信息失败!”信息后,5秒后页面自动刷新到manageFoodType.jsp页面。

3.项目实施

按照项目设计的内容在ch05\WebRoot\admin目录下创建文件manageFoodType.jsp、addFoodType.jsp、addFoodType_do.jsp、editFoodType.jsp、editFoodType_do.jsp和deleteFoodType.jsp。

4.项目运行

启动Tomcat,打开浏览器在地址栏输入URL:http://127.0.0.1:8080/ch05/mealsystem/admin/manageFoodType.jsp,然后依次实验添加、修改和删除功能,观察并记录页面效果图。

【教学效果测评】

本单元将通过课堂提问和课下编程进行效果测评,其中课堂提问详见习题,课下编程详见编程。

1.习题

(1)简述JDBC访问数据库的步骤以及常用的API。

(2)Statement对象可以处理哪些类型的SQL语句?处理这些语句的主要方法是什么?

(3)简述JDBC驱动程序的类型。

2.编程

(1)在MySQL数据库中创建一个数据库ch05,并且在该数据库下创建学生信息表student,含有学生的基本信息。通过JDBC实现对student表中信息的添加、修改、删除和查询操作。

(2)使用JDBC实现网络点餐系统的用户点餐功能。