# 第 5 章

# 数据库访问

**学习目标**

- 理解 JDBC 数据库连接的基本概念和原理
- 掌握应用 JDBC 实现数据库增加、删除、查询、修改功能的编程方法
- 了解数据库访问代码的优化方法

## 5.1 用户注册功能完善

### 5.1.1 注册提交功能设计与实现

**1. 注册提交功能设计**

第 4 章已经描述过用户注册功能的开发任务，并且完成了第一个子任务注册确认功能的开发，这里对第二个子任务注册提交功能进行设计和实现。

注册提交功能的流程控制和数据传递都与用户登录功能类似，共涉及 3 个页面。

首先在图 4.2 所示的注册确认页面中核对注册信息无误后，单击“注册”按钮进入到注册提交页面，同时需要将注册信息也传递过来。在注册提交页面中将注册信息存入数据库的用户数据表中并显示处理结果，最后将执行流程转到用户登录页面。从注册确认页面到注册提交页面的流程控制还需要同步传递数据，因此选择使用 HTML 的表单来实现比较合适。注册提交页面到用户登录页面的流程控制可以使用 HTML 的超链接来实现。

这里存在两个问题需要解决。第一个问题是之前在第 4 章实现注册确认功能时，只是简单地把用户输入的注册信息以普通文本的形式显示在确认页面上。现在实现注册提交功能的话，需要在注册确认页面中再次以表单的形式来存放用户注册信息，以便实现基于表单的数据传递，但这些表单内容无需在页面中再次显示。这个问题可以通过在注册确认页面中增加一个表单用于传递注册信息，并使用 CSS 中隐藏显示的效果来满足需要。第二个问题则是注册提交功能的核心任务，即将用户注册信息保存到数据库中。这需要用到本章的新知识来实现。

本章实现的程序功能是使用数据库来持久保存相关信息。这里选择使用在 Web 应用开发中最流行的 MySQL 作为示例应用程序的数据库环境，读者需要提前安装配置好 MySQL 8，才能继续下面的学习。在安装过程中要为 root 用户设置密码，本书所用的数据库用户为 root，其密码为 mysql。

这里所用到的用户数据表的表结构已经在 1.3 节中给出了。在数据库环境准备好之后，启动 MySQL 命令行客户端，然后按照以下步骤创建用户数据表。

（1）登录 MySQL

输入 MySQL 登录密码，进入命令行界面，如图 5.1 所示。

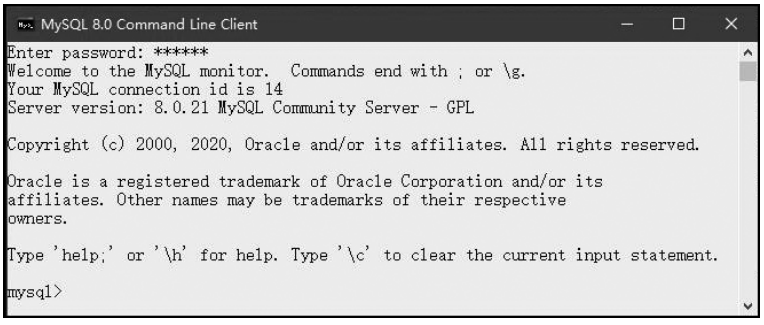


**图 5.1　MySQL 命令行界面**

（2）创建数据库

使用 create database 语句创建示例数据库 newsdb，具体语句如下：

```
create database newsdb
```

（3）创建数据表

先使用 use 语句选择 newsdb 为默认数据库，具体语句如下：

```
use newsdb;
```

然后使用 create table 语句在该数据库中创建数据表 user，具体语句如下：

```
create table user (
    id int(11) NOT NULL AUTO_INCREMENT,
    username varchar(20),
    password varchar(20),
    gender varchar(10),
    resume varchar(100),
    PRIMARY KEY (id)
)
```

创建数据库和数据表的操作过程如图 5.2 所示。

（4）添加管理员用户

使用 insert 语句往 user 数据表中添加一条记录（管理员用户），具体语句如下：

```
insert into user values(1,'admin','admin','male','Administrator');
```

```
mysql> create database newsdb;
Query OK, 1 row affected (0.03 sec)

mysql> use newsdb;
Database changed
mysql> create table user(
    -> id int(11) NOT NULL AUTO_INCREMENT,
    -> username varchar(20),
    -> password varchar(20),
    -> gender varchar(10),
    -> resume varchar(100),
    -> PRIMARY KEY(id)
    -> );
Query OK, 0 rows affected, 1 warning (0.04 sec)
```

**图 5.2　创建数据库和数据表**

添加管理员用户的操作过程如图 5.3 所示。

```
mysql> insert into user values(1,'admin','admin','male','Administrator');
Query OK, 1 row affected (0.01 sec)
```

**图 5.3　添加管理员用户**

(5) 查看 user 数据表结构和内容

先使用 desc 语句查看 user 表结构,具体语句如下:

```
descuser;
```

查看 user 数据表结构的操作过程如图 5.4 所示。

```
mysql> desc user;
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| id       | int          | NO   | PRI | NULL    | auto_increment |
| username | varchar(20)  | YES  |     | NULL    |                |
| password | varchar(20)  | YES  |     | NULL    |                |
| gender   | varchar(10)  | YES  |     | NULL    |                |
| resume   | varchar(100) | YES  |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
5 rows in set (0.02 sec)
```

**图 5.4　查看 user 数据表结构**

然后再使用 select 语句查看 user 表的内容,具体语句如下:

```
select * from user;
```

查看 user 数据表内容的操作过程如图 5.5 所示。

**2. 注册提交功能实现**

**【程序 5.1】** 修改程序 4.3 得到新的注册确认页面文件 userVeriRegister.jsp,在文件中增加表单,用于传递注册信息(黑体部分为新增代码)。

```
mysql> select * from user;
+----+----------+----------+--------+---------------+
| id | username | password | gender | resume        |
+----+----------+----------+--------+---------------+
|  1 | admin    | admin    | male   | Administrator |
+----+----------+----------+--------+---------------+
1 row in set (0.00 sec)
```

图 5.5　查看 user 数据表内容

```
…
<!--main begin -->
<section class="verifyform">
    <div class="top-bar">
        <h1>注册确认</h1>
    </div><br/>

<%request.setCharacterEncoding("utf-8");
    String uName =request.getParameter("username");
    String uPwd =request.getParameter("password");
    String uGender =request.getParameter("gender");
    String uResume =request.getParameter("resume");
%>
<table>
    <tr>
        <td>用户名: </td><td><%=uName %></td>
    </tr>
    <tr>
        <td>密码: </td><td><%=uPwd %></td>
    </tr>
    <tr>
        <td>性别: </td><td><%=uGender %></td>
    </tr>
    <tr>
        <td>个人简介:</td><td><%=uResume %></td>
    </tr>
</table>

<form action="doRegister.jsp" method="post">
    <div style="display:none">
        用户名:
        <input type="text" name="username" value="<%=uName%>"/><br/>
```

```
            密    码:
            <input type="text" name="password" value="<%=uPwd %>"/><br/>
            性    别:
            <input type="text" name="gender" value="<%=uGender%>"/><br/>
            个人简介<br>
            <textarea name="resume" cols="20" rows="6"><%=uResume%></textarea>
            <br><br>           
        </div>
        <input class="btn" type="submit" value="注册"></input>
        <a href="javascript:history.go(-1)" >返回</a><br/>
    </form>
    </section>
    <!--main end -->
    …
```

第4章中的userVeriRegister.jsp已经实现了获取表单数据并在页面中显示的功能，这里增加了一个表单用于继续往下一页面传递用户的注册信息，可以使用HTML的form标记和input标记来实现。

form标记中的action属性取值为“doRegister.jsp”，实现到注册提交页面的流程控制。表单中的内容用input标记和textarea标记来分别定义用户名、密码、性别和个人简介。因为在用户注册页面中已经输入了这些信息，这里直接使用之前已经获取到的相应值设置各个input标记的value属性值及textarea标记的内容。

另外，用input标记实现了“注册”提交按钮，用a标记实现了“返回”超链接。当单击“注册”按钮时程序流程转向注册提交页面并同步传递表单中的数据，单击“返回”超链接则返回到用户注册页面，从而可以修改注册信息之后再重新提交。

因为在页面中已经用表格形式显示了用户注册信息供核对，表单中要提交的注册信息不需要再显示，所以将这部分内容放置在一个div中，并设置其CSS样式的display属性值为none。这样程序在运行时会将相关内容隐藏起来，避免相同内容在页面上重复出现。

**【程序5.2】** 在newsPub文件夹下编写注册提交页面文件doRegister.jsp，将用户注册信息写入数据库。

```
<%@page language="java" contentType="text/html;charset=utf-8"
pageEncoding="utf-8"%>
<!--1. 导入 JDBC API -->
<%@page import="java.sql.*" %>

<%request.setCharacterEncoding("utf-8");
```

```
String uname =request.getParameter("username");
String pwd =request.getParameter("password");
String gender =request.getParameter("gender");
String resume =request.getParameter("resume");

Connection conn =null;
Statement stmt =null;
String sDBDriver ="com.mysql.cj.jdbc.Driver";
String sConnStr ="jdbc:mysql://localhost:3306/newsdb?useUnicode =
 true&characterEncoding=utf-8";
String username ="root";
String password ="mysql";

//2.装载驱动程序
try {
    Class.forName(sDBDriver);
}
catch(ClassNotFoundException ex) {
    System.err.println(ex.getMessage());
}

try {
    //3.创建数据库连接
    conn =DriverManager.getConnection(sConnStr,username,password);

    //4.创建 Statement 对象
    stmt =conn.createStatement();

    //5.执行 SQL 语句
    String sql="INSERT INTO user VALUES (null,'"+uname+"','"+
            pwd+"','"+gender+"','"+resume+"')";
    int result =stmt.executeUpdate(sql);

    //6.处理结果
    if(result==1){
        out.println("用户注册成功,请登录使用。");
    }else{
        out.println("用户注册失败,请联系管理员!");
    }
    out.println("<a href='userLogin.jsp'>登录</a>");
}
catch(SQLException e1) { out.println(e1);}
finally{
```

```
            //7.关闭连接
            conn.close();
        }
    %>
```

doRegister.jsp 负责获取表单数据并写入数据库。获取表单数据的处理方法与注册确认页面中完全相同，这里不再赘述。下面解释一下将数据写入到数据库所用的代码。

首先用 page 指令导入 java.sql 包。这个包具有访问数据库所需的一些类。

然后定义数据库操作需要用到的参数，包括驱动程序类名 sDBDriver、数据库连接字符串 sConnStr、数据库访问用户名 username 以及数据库访问密码 password。

接下来使用 Class 类的 forName()方法注册数据库驱动程序，再使用 DriverManager 的 getConnection()方法创建一个数据库连接，然后用 Connection 对象的 createStatement()方法创建一个 Statement 对象。

之后使用获取的用户注册信息拼接成一条 INSERT 语句，并调用 Statement 对象的 executeUpdate()方法来执行它。这条 SQL 语句会通过 JDBC 发送给数据库系统执行，并返回执行的结果。当 INSERT 语句执行成功后，会在数据表中新增一条记录，并返回影响的记录条数，这里应该为整数 1。

再根据返回值输出注册结果提示信息，并给出一个转到用户登录页面的超链接以实现到下一步的流程控制。

最后关闭数据库连接，释放所占用的资源。

由于上面使用的一些方法在执行过程中有可能抛出异常，因此需要将这些代码分别放置在不同的 try 语句块中。

使用程序 4.1 作为用户登录页面 userLogin.jsp，使用程序 4.2 作为用户注册页面 userRegister.jsp。然后再从 https://dev.mysql.com/downloads/file/?id=496589 下载 MySQL 8 的 JDBC 驱动并解压缩。将解压得到的驱动文件 mysql-connector-java-8.0.21.jar 复制到 newsPub\WEB-INF\lib 文件夹下，完成注册提交功能的开发。

### 5.1.2　注册提交功能运行过程

用户注册功能的程序运行过程详述如下。

启动 Tomcat，打开浏览器，在地址栏输入 http://localhost:8080/newsPub/userLogin.jsp，出现用户登录页面，单击“用户注册”超链接，进入用户注册页面。

在注册表单中输入用户注册信息后，单击“提交”按钮，注册表单被提交给注册确认页面，显示效果如图 5.6 所示。

注册确认页面中的注册信息核对如果有问题，可以单击“返回”回到用户注册页面重新输入。如果核对无误可以单击“注册”按钮，注册确认页面中隐藏的确认表单被提交给注册提交页面。

注册提交页面中的代码运行将用户注册信息写入数据库，实现用户注册功能。完成后显示注册成功提示，如图 5.7 所示。

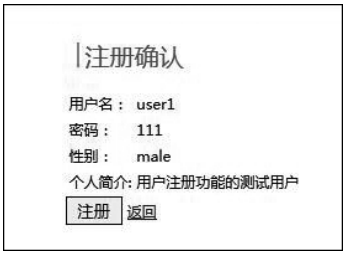


图 5.6 注册确认页面

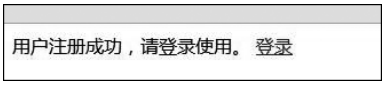


图 5.7 注册成功提示页面

最后可以单击“登录”链接回到用户登录页面。

### 5.1.3 数据库访问代码模板

Java 程序中对数据库的访问可以基于 JDBC 实现。编写数据库访问代码遵循相对固定的模式。这里以操作 MySQL 数据库为例，给出实现数据库访问的典型代码模板，主要包括以下步骤。

**1. 导入 JDBC API**

首先用 page 指令导入 java.sql 包。

```
<%@page import="java.sql.*" %>
```

**2. 加载驱动程序**

然后使用 Class 类的 forName()方法加载驱动程序类，加载成功后即可使用该驱动程序与数据库建立连接。

```
Class.forName("com.mysql.cj.jdbc.Driver");
```

**3. 建立数据库连接**

提供数据库连接字符串、数据库用户名和数据库密码，使用 DriverManager 类的 getConnection()方法建立数据库连接。

```
String sConnStr ="jdbc:mysql://localhost:3306/newsdb?useUnicode =
 true&characterEncoding=utf-8";
String username ="root";
String password ="mysql";
Connection conn = DriverManager. getConnection ( sConnStr, username,
password);
```

**4. 创建 Statement 对象**

使用数据库连接对象的 createStatement()方法创建 Statement 对象。

```
stmt =conn.createStatement();
```

**5. 执行 SQL 语句**

使用 Statement 对象的 executeUpdate()方法执行 INSERT 语句。

```
String sql="INSERT INTO user VALUES(null,'"+uname+"','"+
            pwd+"','"+gender+"','"+resume+"')";
int result =stmt.executeUpdate(sql);
```

**6. 处理执行结果**

根据需要对执行结果 result 进行处理。

```
if(result ==1){
        out.println("用户注册成功,请登录使用。");
    }else{
    out.println("用户注册失败,请联系管理员!");
}
```

**7. 关闭链接**

使用 close()方法关闭数据库连接,释放所占用的资源。

```
conn.close();
```

在注册提交页面中就是按照这个步骤实现了往数据库中写入用户注册信息的功能。

需要说明的是,由于调用的这些方法在执行过程中有可能抛出异常,因此需要将这些代码放置在 try 语句块中。程序最后必须关闭链接以释放资源,所以要把相关代码放在 finally 语句块中,以确保在程序运行出现异常时数据库连接也能关闭。

另外需要注意的是,必须提前创建好相关的数据库、数据表,并配置好相应的数据库驱动程序。

掌握了这个代码模板之后,如果需要实现对数据库进行其他类型的操作,都可以参考它编写程序来实现。

## 5.2　用户登录功能完善

### 5.2.1　登录判断功能设计与实现

**1. 登录判断功能设计**

前面在第 4 章已经描述过用户登录功能的开发任务,并基本完成了它的设计和实现。但是在实现登录判断功能时,是用固定的用户账号来模拟实现的。实际的用户身份判断应当根据数据库中保存的用户名和密码等信息来实现。

如果用户登录时输入的用户名和密码在数据库中存有相应的记录，就证明该用户是合法用户，可以允许登录，否则登录失败。这里利用 JDBC 来实现完整的登录判断功能。

**2. 登录判断功能实现**

**【程序 5.3】** 修改程序 4.4 得到新的登录判断页面文件 doLogin.jsp，把文件中原来使用固定的用户账号信息模拟实现用户身份判断，改为根据数据库中的真实用户信息来判断。

```
<%@page language="java" contentType="text/html;charset=utf-8"%>
<!--1. 导入 JDBC API -->
<%@page import="java.sql.* " %>

<%
    request.setCharacterEncoding("utf-8");
    String uname =request.getParameter("username");
    String pwd =request.getParameter("password");

    Connection conn =null;
    ResultSet rs =null;
    Statement stmt =null;

    String sDBDriver ="com.mysql.cj.jdbc.Driver";
    String sConnStr ="jdbc:mysql://localhost:3306/newsdb?useUnicode =
      true&characterEncoding=utf-8";
    String username ="root";
    String password ="mysql";

    //2.装载驱动程序
    try {
        Class.forName(sDBDriver);
    }
    catch(ClassNotFoundException ex) {
          System.err.println(ex.getMessage());
    }

    try {
        //3.创建数据库连接
        conn =DriverManager.getConnection(sConnStr,username,password);

        //4.创建 Statement 对象
        stmt =conn.createStatement();
```