**实验九 JDBC访问数据库**

一、实验目的

1. 掌握使用传统的方法访问数据库；

2. 掌握使用数据源的方法访问数据库。

二、实验原理

数据库应用是Web应用开发的一个重要应用。Web应用程序访问数据库有两种方法：传统的方法和使用JNDI数据源的方法。

传统方法访问数据库的步骤是：①加载数据库驱动程序；②建立连接对象；③创建语句对象；④获得结果集；⑤关闭有关连接对象。

使用数据源访问数据库的步骤是：①配置数据源（局部数据源或全局数据源）；②通过JNDI机制查找命名数据源；③通过数据源对象创建连接对象；④其他与传统方法一致。

三、实验内容与步骤

**1、使用传统方法通过JSP页面访问数据库**

【步骤1】创建数据库。在MySQL建立了一个名为bookstore的数据库，在其中建立books表，代码如下：

CREATE TABLE IF NOT EXISTS `books` (

`bookid` varchar(64) NOT NULL,

`title` varchar(64) NOT NULL,

`author` varchar(64) NOT NULL,

`publisher` varchar(64) NOT NULL,

`price` double(10,2) DEFAULT NULL,

PRIMARY KEY (`bookid`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

向bookstore表中插入几条记录，代码如下：

INSERT INTO `books` VALUES ('204','Head First Servlets & JSP', 'Bryan Basham', '中国电力出版社',98.00);

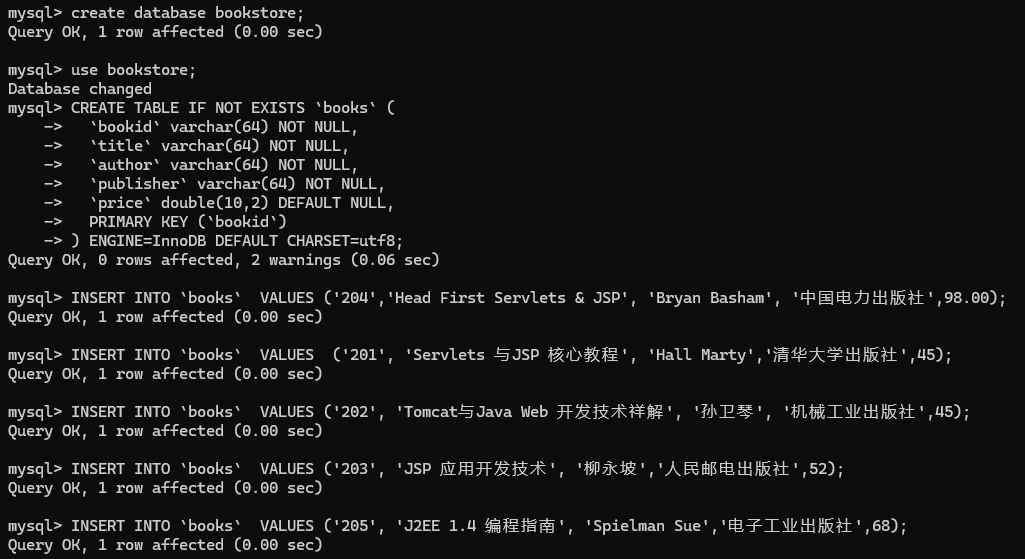
INSERT INTO `books` VALUES ('201', 'Servlets 与JSP 核心教程', 'Hall Marty','清华大学出版社',45);

INSERT INTO `books` VALUES ('202', 'Tomcat与Java Web 开发技术祥解', '孙卫琴', '机械工业出版社',45);

INSERT INTO `books` VALUES ('203', 'JSP 应用开发技术', '柳永坡','人民邮电出版社',52);

INSERT INTO `books` VALUES ('205', 'J2EE 1.4 编程指南', 'Spielman Sue','电子工业出版社',68);

1. 运行截图



注意：需要将数据库的JDBC驱动程序安装到应用程序的WEB-INF\lib目录中。

【步骤2】完成JSP页面displayBooks.jsp，实现访问books表中的数据。



图9-1 displayBooks.jsp的运行效果

1. 代码（10’）

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

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>display</title>

</head>

<body>

<%

try {

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

String dburl = "jdbc:mysql://127.0.0.1:3306/bookstore";

String username = "root";

String password = "";

Connection conn = DriverManager.getConnection(dburl, username, password);

if (conn != null) {

Statement stmt = conn.createStatement();

String sql = "select \* from books";

ResultSet rs = stmt.executeQuery(sql);

out.print("<table border ='1'>");

out.print("<tr><td>书号</td><td>书名</td><td>作者</td><td>价格</td></tr>");

while (rs.next()) {

out.print("<tr><td>" + rs.getString(1) + "</td><td>" + rs.getString(2) + "</td><td>"

+ rs.getString(3) + "</td><td>" + rs.getDouble(5) + "</td></tr>");

}

out.print("</table>");

rs.close();

stmt.close();

conn.close();

}

} catch (Exception e) {

e.printStackTrace();

}

%>

</body>

</html>

**2、通过数据源访问数据库**

注意：需要将数据库的JDBC驱动程序安装到Tomcat安装目录的\lib目录中,并重新启动Tomcat服务器。

【步骤1】建立局部数据源

在Web应用程序中建立一个META-INF目录，在其中建立一个context.xml文件。

（1）代码（10’）

<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/2002/xmlspec/dtd/2.10/xmlspec.dtd">

<Context reloadable = "true">

<Resource

name = "DataSource"

type = "javax.sql.DataSource"

maxActive = "10"

maxIdle = "4"

username = "root"

password = ""

maxWait = "5000"

driverClassName = "com.mysql.jdbc.Driver"

url = "jdbc:mysql://127.0.0.1:3306/bookstore"/>

</Context>

【步骤2】完成JSP页面displayBooks.jsp，实现访问表books的数据。



图9-2 displayBooks.jsp的运行效果

（2）代码（10’）

<%@page import="java.sql.\*,javax.naming.\*,javax.sql.\*"%>

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>display2</title>

</head>

<body>

<%

try {

Context context = new InitialContext();

DataSource ds = (DataSource) context.lookup("java:comp/env/DataSource");

Connection conn = ds.getConnection();

if (conn != null) {

Statement stmt = conn.createStatement();

String sql = "select \* from books";

ResultSet rs = stmt.executeQuery(sql);

out.print("<table border ='1'>");

out.print("<tr><td>书号</td><td>书名</td><td>作者</td><td>价格</td></tr>");

while (rs.next()) {

out.print("<tr><td>" + rs.getString(1) + "</td><td>" + rs.getString(2) + "</td><td>"

+ rs.getString(3) + "</td><td>" + rs.getDouble(5) + "</td></tr>");

}

out.print("</table>");

rs.close();

stmt.close();

conn.close();

}

} catch (Exception e) {

e.printStackTrace();

}

%>

</body>

</html>

**3、综合应用**

本实验采用MVC设计模式，通过数据源和DAO对象访问数据库。其中JavaBeans实现模型，访问数据库，Servlet实现控制器，JSP页面实现视图。

模型包括2个JavaBean：BookBean用于存放图书信息，BookDAO用于访问数据库。

控制器包括2个Servlet：BookQueryServlet根据请求参数查询图书信息、BookInsertServlet用来向数据库中插入数据。

视图包括4个JSP页面：bookQuery.jsp显示查询页面、bookInsert.jsp显示插入页面、display.jsp显示查询结果页面和errorPage.jsp显示错误页面。

【步骤1】完成BookBean.java，实现用于存放图书信息的JavaBeans代码。

（1）代码（5’）

package com.demo;

public class BookBean {

private String bookid;

private String title;

private String author;

private String publisher;

private double price;

public BookBean() {

super();

}

public BookBean(String bookid, String title, String author, String publisher, double price) {

super();

this.bookid = bookid;

this.title = title;

this.author = author;

this.publisher = publisher;

this.price = price;

}

public String getBookid() {

return bookid;

}

public void setBookid(String bookid) {

this.bookid = bookid;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getAuthor() {

return author;

}

public void setAuthor(String author) {

this.author = author;

}

public String getPublisher() {

return publisher;

}

public void setPublisher(String publisher) {

this.publisher = publisher;

}

public double getPrice() {

return price;

}

public void setPrice(double price) {

this.price = price;

}

}

【步骤2】完成BookDAO，作为一个简单的JavaBeans，实现数据库的访问功能、根据书号查询图书信息功能以及插入图书记录的功能。

（2）代码（10’）

package com.demo;

import java.sql.\*;

import javax.naming.InitialContext;

import javax.naming.NamingException;

import javax.sql.DataSource;

public class BookDAO {

private static InitialContext context = null;

private DataSource dataSource = null;

public BookDAO() {

try {

if (context == null) {

context = new InitialContext();

}

dataSource = (DataSource) context.lookup("java:comp/env/jdbc/bookDS");

} catch (NamingException e2) {

}

}

public BookBean searchBook(String bookid) throws SQLException {

Connection conn = null;

PreparedStatement pstmt = null;

ResultSet rs = null;

try {

conn = dataSource.getConnection();

String sql = "select \* from boos where bookid=?";

pstmt = conn.prepareStatement(sql);

pstmt.setString(1, bookid);

rs = pstmt.executeQuery();

BookBean book = new BookBean();

if (rs.next()) {

book.setBookid(rs.getString(1));

book.setTitle(rs.getString(2));

book.setAuthor(rs.getString(3));

book.setPublisher(rs.getString(4));

book.setPrice(rs.getDouble(5));

}

return book;

} catch (Exception e) {

return null;

} finally {

rs.close();

pstmt.close();

conn.close();

}

}

public boolean insertBook(BookBean book) throws SQLException {

Connection conn = null;

PreparedStatement pstmt = null;

try {

conn = dataSource.getConnection();

String sql = "insert into books values (?,?,?,?,?)";

pstmt = conn.prepareStatement(sql);

pstmt.setString(1, book.getBookid());

pstmt.setString(2, book.getTitle());

pstmt.setString(3, book.getAuthor());

pstmt.setString(4, book.getPublisher());

pstmt.setDouble(5, book.getPrice());

int n = pstmt.executeUpdate();

if (n == 1) {

return true;

} else {

return false;

}

} catch (Exception e) {

return false;

} finally {

pstmt.close();

conn.close();

}

}

}

【步骤3】完成bookQuery.jsp，实现根据书号查询图书信息。

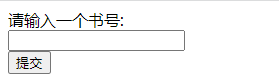


图9-3 bookQuery.jsp的运行效果

1. 代码（10’）

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>bookQuery</title>

</head>

<body>

请输入一个书号:

<br />

<form action="../BookQueryServlet" method="post">

<input name="bookid"><br /> <input type="submit" value="提交">

</form>

</body>

</html>

【步骤4】完成bookInsert.jsp，实现将用户输入的图书信息插入到数据库中。

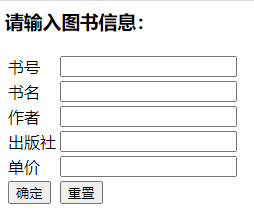


图9-4 bookInsert.jsp的运行截图

（4）代码（10’）

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>bookInsert</title>

</head>

<body>

<h3>请输入图书信息：</h3>

${result}

<form action="../BookInsertServlet" method="post">

<table>

<tr>

<td>书号</td>

<td><input name="bookid"></td>

</tr>

<tr>

<td>书名</td>

<td><input name="title"></td>

</tr>

<tr>

<td>作者</td>

<td><input name="author"></td>

</tr>

<tr>

<td>出版社</td>

<td><input name="publisher"></td>

</tr>

<tr>

<td>单价</td>

<td><input name="price"></td>

</tr>

<tr>

<td><input type="submit" value="确定"></td>

<td><input type="reset" value="重置"></td>

</tr>

</table>

</form>

</body>

</html>

【步骤5】完成display.jsp，实现显示查询结果。



图9-5 display.jsp的运行效果

（5）代码

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<jsp:useBean id="book" class="com.demo.BookBean" scope="request" />

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>display</title>

</head>

<body>

书号：<jsp:getProperty name="book" property="bookid" />

书名：<jsp:getProperty name="book" property="title" />

作者：<jsp:getProperty name="book" property="author" />

出版社：<jsp:getProperty name="book" property="publisher" />

价格：<jsp:getProperty name="book" property="price" />

</body>

</html>

【步骤6】完成错误页面errorPage.jsp，实现提示用户所查图书不存在。



图9-6 errorPage.jsp的运行效果

（6）代码（10’）

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>errorPage</title>

</head>

<body>

对不起，您查的图书不存在！

</body>

</html>

【步骤7】完成BookQueryServlet，实现从请求参数获得书号，然后从数据库中查找该书，最后根据查询结果将请求转发到显示页面(display.jsp)或错误页面(errorPage.jsp)。

（7）代码（10’）

package com.demo;

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet("/BookQueryServlet")

public class BookQueryServlet extends HttpServlet {

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

String bookid = request.getParameter("bookid");

BookDAOImplement queryBook = new BookDAOImplement();

BookBean book = new BookBean();

try {

book = queryBook.searchBook(bookid);

if (book != null) {

request.setAttribute("book", book);

request.getRequestDispatcher("display.jsp").forward(request, response);

} else {

request.getRequestDispatcher("errorPage.jsp").forward(request, response);

}

} catch (Exception e) {

e.printStackTrace();

}

}

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

doGet(request, response);

}

}

【步骤8】完成BookInsertServlet ，实现向数据库插入数据，并将控制请求的转发到bookInsert.jsp页面。

（8）代码（10’）

package com.demo;

import java.io.IOException;

import java.sql.SQLException;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import com.demo.BokDAO;

@WebServlet("/BookInsertServlet")

public class BookInsertServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

request.setCharacterEncoding("gb2312");

String message = null;

BookBean book = new BookBean(request.getParameter("bookid"), request.getParameter("title"),

request.getParameter("author"), request.getParameter("publisher"),

Float.parseFloat(request.getParameter("price")));

BokDAO bookdao = new BookDAO();

try {

boolean success = bookdao.insertBook(book);

if (success) {

message = "成功插入一条记录！";

} else {

message = "插入记录错误！";

}

} catch (SQLException e) {

e.printStackTrace();

}

request.setAttribute("result", message);

RequestDispatcher view = request.getRequestDispatcher("/bookInsert.jsp");

view.forward(request, response);

}

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

doGet(request, response);

}

}

【步骤9】在DD文件中部署Servlet。

（9）代码（5’）

<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd" id="WebApp\_ID" version="3.1">

<display-name>ch9</display-name>

<welcome-file-list>

<welcome-file>index.html</welcome-file>

<welcome-file>index.htm</welcome-file>

<welcome-file>index.jsp</welcome-file>

<welcome-file>default.html</welcome-file>

<welcome-file>default.htm</welcome-file>

<welcome-file>default.jsp</welcome-file>

</welcome-file-list>

<resource-ref>

<description>MySQL DBCP</description>

<res-ref-name>DataSource</res-ref-name>

<res-type>javax.sql.DataSource</res-type>

<res-auth>Container</res-auth>

</resource-ref>

<servlet>

<servlet-name>bookQuery</servlet-name>

<servlet-class>com.demo.BookQueryServlet</servlet-class>

</servlet>

<servlet>

<servlet-name>bookInsert</servlet-name>

<servlet-class>com.demo.BookInsertServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>bookQuery</servlet-name>

<url-pattern>/bookquery.do</url-pattern>

</servlet-mapping>

<servlet-mapping>

<servlet-name>bookInsert</servlet-name>

<url-pattern>/bookinsert.do</url-pattern>

</servlet-mapping>

</web-app>