**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。（10分）



图9-1 displayBooks.jsp的运行效果

1. 代码

<%@ page contentType="text/html; charset=gb2312"%>

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

<html>

<head>

<title>Database Access Test</title>

</head>

<body>

<%

try {

**Class.forName("com.mysql.cj.jdbc.Driver");【1分】**

**String dburl = "jdbc:mysql://localhost:3307/bookstore?serverTimezone=UTC";【1分】**

**Connection conn = DriverManager.getConnection(dburl, "root", "123456");【1分】**

**Statement stmt = conn.createStatement();【1分】**

**String sql = "SELECT \* FROM books";【1分】**

**ResultSet rs = stmt.executeQuery(sql); 【1分】**

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

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

while (rs.next()) {

**out.println("<tr><td>" + rs.getString(1) + "</td><td>" + rs.getString(2) + "</td><td>" + rs.getString(3)**

**+ "</td><td>" + rs.getString(5) + "</td></tr>");【2分】**

}

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

**rs.close();**

**stmt.close();**

**conn.close();【2分】**

} catch (Exception e) {

out.println(e.getMessage());

}

%>

</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（5分）

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

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

**<Context reloadable="true">【1分】**

**<Resource**

**name = "jdbc/bookDS" 【1分】**

**type = "javax.sql.DataSource"【1分】**

**maxActive = "4" 【1分】**

**maxIdle = "2" 【1分】**

**driverClassName = "com.mysql.cj.jdbc.Driver" 【1分】**

**url = "jdbc:mysql://localhost:3307/bookstore"【1分】**

**username = "root" 【1分】**

**password = "123456" 【1分】**

**maxWaitMillis = "5000" />【1分】**

**</Context>**

【步骤2】完成JSP页面displayBooks.jsp，实现访问表books的数据。（10分）



图9-2 displayBooks.jsp的运行效果

（2）代码

<%@ page contentType="text/html; charset=gb2312"%>

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

<html>

<head>

<title>DataSource Test</title>

</head>

<body>

<%

try {

**Context context = new InitialContext();【1分】**

**DataSource ds = (DataSource) context.lookup("java:comp/env/jdbc/bookDS");【1分】**

**Connection conn = ds.getConnection();【1分】**

**Statement stmt = conn.createStatement();【1分】**

**ResultSet rs = stmt.executeQuery("SELECT \* FROM books");【1分】**

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

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

**while (rs.next()) {**

**out.println("<tr><td>" + rs.getString(1) + "</td><td>" + rs.getString(2) + "</td><td>" + rs.getString(3)**

**+ "</td><td>" + rs.getString(5) + "</td></tr>");【2分】**

}

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

**rs.close();**

**stmt.close();**

**conn.close();【2分】**

} catch (Exception e) {

out.println(e.getMessage());

}

%>

</body>

</html>

3、综合应用（100）

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。（5分）

（1）代码

package com.beans;

import java.io.\*;

**public class BookBean implements Serializable {【1分】**

private String bookid = null;

private String title = null;

private String author = null;

private String publisher = null;

private float price = 0.0F;

**public BookBean() {}【1分】**

public BookBean(String bookId, String author, String title, String publisher, float price) {

this.bookid = bookId;

this.title = title;

this.author = author;

this.publisher = publisher;

this.price = price;

}

public String getBookid() {

return this.bookid;

}

public String getTitle() {

return title;

}

public String getAuthor() {

return this.author;

}

public float getPrice() {

return price;

}

public String getPublisher() {

return publisher;

}

public void setBookid(String bookid) {

this.bookid = bookid;

}

public void setTitle(String title) {

this.title = title;

}

public void setAuthor(String author) {

this.author = author;

}

public void setPrice(float price) {

this.price = price;

}

public void setPublisher(String publisher) {

this.publisher = publisher;

}

}

【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。

DAO（不占分）： DAOE（不占分）： BookDAO（不占分） BookDAOImpl

DAO：（不占分）

package com.dao;

import java.sql.\*;

import javax.sql.DataSource;

import javax.naming.\*;

public interface DAO {

// 查找并返回数据源对象

public static DataSource getDataSource() {

DataSource dataSource = null;

try {

Context context = new InitialContext();

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

} catch (NamingException ne) {

System.out.println("异常:" + ne);

}

return dataSource;

}

// 返回连接对象方法

public default Connection getConnection() throws DaoException {

DataSource dataSource = getDataSource();

Connection conn = null;

try {

conn = dataSource.getConnection();

} catch (SQLException sqle) {

System.out.println("异常:" + sqle);

}

return conn;

}

}

1. BookDAO 代码

package com.dao;

import java.util.ArrayList;

import com.model.Customer;

public interface BookDAO extends DAO {

public boolean insertBook(BookBean book) throws DaoException;

public BookBean searchBook(String bookid) throws DaoException;

}

**public class CustomerDaoImpl implements CustomerDao {**

**// 根据书号查询图书信息**

**public BookBean searchBook(String bookid) {【5分】**

**Connection conn = null;**

**PreparedStatement pstmt = null;**

**ResultSet rst = null;**

**BookBean book = new BookBean();**

**try {**

**conn = getConnection();**

**pstmt = conn.prepareStatement("SELECT \* FROM books WHERE bookid=?");**

**pstmt.setString(1, bookid);**

**rst = pstmt.executeQuery();**

**if (rst.next()) {**

**book.setBookid(rst.getString("bookid"));**

**book.setTitle(rst.getString("title"));**

**book.setAuthor(rst.getString("author"));**

**book.setPublisher(rst.getString("publisher"));**

**book.setPrice(rst.getFloat("price"));**

**return book;**

**} else {**

**return null;**

**}**

**} catch (SQLException se) {**

**return null;**

**} finally {**

**try {**

**conn.close();**

**} catch (SQLException se) {**

**}**

**}**

**}**

**// 插入一本图书记录**

**public boolean insertBook(BookBean book) {【5分】**

**Connection conn = null;**

**PreparedStatement pstmt = null;**

**try {**

**conn = dataSource.getConnection();**

**pstmt = conn.prepareStatement("INSERT INTO books VALUES(?,?,?,?,?)");**

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

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

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

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

**pstmt.setFloat(5, book.getPrice());**

**pstmt.executeUpdate();**

**pstmt.close();**

**return true;**

**} catch (SQLException se) {**

**return false;**

**} finally {**

**try {**

**conn.close();**

**} catch (SQLException se) {**

**}**

**}**

**}**

**}**

package com.dao;

public class DaoException extends Exception {

private static final long serialVersionUID = 19192L;

private String message;

public DaoException() {

}

public DaoException(String message) {

this.message = message;

}

public String getMessage() {

return message;

}

public void setMessage(String message) {

this.message = message;

}

public String toString() {

return message;

}

}

【步骤3】完成bookQuery.jsp，实现根据书号查询图书信息。（5分）

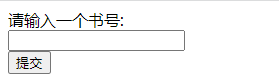


图9-3 bookQuery.jsp的运行效果

（3）代码

<%@ page contentType="text/html; charset=gb2312"%>

<html>

<head>

<title>Book Query</title>

</head>

<body>

请输入一个书号:

<br>

**<form action="bookquery.do" method="post">【2分】**

**<input type="text" name="bookid"><br> <input**

**type="submit" value="提交">【2分】**

**</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。（5分）

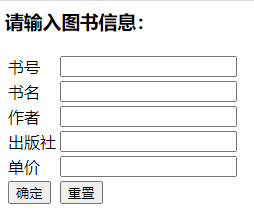


图9-4 bookInsert.jsp的运行截图

（4）代码

<%@ page contentType="text/html; charset=gb2312"%>

<html>

<head>

<title>Book Insert</title>

</head>

<body>

<h3>请输入图书信息：</h3>

<%

if (request.getAttribute("result") != null) **【1分】**

out.print(request.getAttribute("result"));**【1分】**

%>

<form action="bookinsert.do" method="post">**【2分】**

<table>

<tr>

<td>书号</td>

<td><input type="text" name="bookid"></td>**【1分】**

</tr>

<tr>

<td>书名</td>

<td><input type="text" name="title"></td>**【1分】**

</tr>

<tr>

<td>作者</td>

<td><input type="text" name="author"></td>**【1分】**

</tr>

<tr>

<td>出版社</td>

<td><input type="text" name="publisher"></td>**【1分】**

</tr>

<tr>

<td>单价</td>

<td><input type="text" name="price"></td>**【1分】**

</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 contentType="text/html;charset=gb2312"%>

<jsp:useBean id="book" class="com.beans.BookBean" scope="request" />**【2分】**

<jsp:setProperty name="book" property="\*" />**【1分】**

<html>

<body>

书号：<jsp:getProperty name="book" property="bookid" />**【1分】**

书名：<jsp:getProperty name="book" property="title" />**【1分】**

作者：<jsp:getProperty name="book" property="author" />**【1分】**

出版社：<jsp:getProperty name="book" property="publisher" />**【1分】**

价格：<jsp:getProperty name="book" property="price" />**【1分】**

</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

<%@ page contentType="text/html;charset=gb2312"%>

<html>

<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)。（10分）

（7）代码

package com.control;

import java.io.\*;

import java.sql.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

import com.beans.BookBean;

import com.beans.BookDAO;

public class BookQueryServlet extends HttpServlet{

public void doPost(HttpServletRequest request,HttpServletResponse response)

throws ServletException,IOException{

String bookid = request.getParameter("bookid");**【1分】**

BookDAO bookdao = new BookDAO();**【1分】**

BookBean book = bookdao.searchBook(bookid); **【2分】**

if(book!=null){

request.getSession().setAttribute("book", book); **【1分】**

RequestDispatcher view = request.getRequestDispatcher("/display.jsp");**【1分】**

view.forward(request, response);

}else{

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

view.forward(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。（10分）

（8）代码

package com.control;

import java.io.\*;

import java.sql.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

import com.beans.BookBean;

import com.beans.BookDAO;

public class BookInsertServlet extends HttpServlet {

public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

request.setCharacterEncoding("gb2312");

String message = null;

BookBean book = new BookBean(request.getParameter("bookid"), request.getParameter("title"),**【2分】**

request.getParameter("author"), request.getParameter("publisher"),**【2分】**

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

BookDAO bookdao = new BookDAO();**【2分】**

boolean success = bookdao.insertBook(book); **【2分】**

if (success) {

message = "成功插入一条记录！";

} else {

message = "插入记录错误！";

}

request.setAttribute("result", message);

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

view.forward(request, response);

}

}

【步骤9】在DD文件中部署Servlet。（5分）

（9）代码

<servlet>

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

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

</servlet>

<servlet>

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

<servlet-class>com.control.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>