### JAVA第二阶段—DAY04-JAVA作业答案

1. 简述什么是JDBC以及JDBC的作用

什么是JDBC：Java DataBase Connectivity(Java数据库连接) JDBC是Java访问数据库的标准规范

JDBC的作用：JDBC是用于执行SQL语句的Java API(Java语言通过JDBC可以操作数据库)

2、说明一下JDBC操作的4个基本对象分别是什么

DriverManager: 用于注册驱动

Connection: 表示与数据库创建的连接

Statement: 执行SQL语句的对象

ResultSet: 结果集或一张虚拟表

3、说明一下jdbc普通语句对象与预处理语句对象的区别

平常的Statement每次执行SQL，数据库管理系统（DBMS）都会解析编译SQL成可执行代码，然后执行并返回结果。 ​ 预处理语句，就是将SQL中可变的参数抽取出来替换，数据库管理系统（DBMS）会把SQL不变的部分编译成可执行代码，每次运行时，将参数代入代码，直接执行。

在需要反复执行特定SQL时，使用预处理语句的优势是：

预处理语句大大减少了分析时间，只做了一次查询（虽然语句多次执行）。

绑定参数减少了服务器带宽，你只需要发送查询的参数，而不是整个语句。

预处理语句针对SQL注入是非常有用的，因为参数值发送后使用不同的协议，保证了数据的合法性。

4、编程题

在数据库中建2张表，员工表和打卡信息表

-- 员工表

create table emp(

empno int auto\_increment, -- 工号

ename varchar(200) default '' not null, -- 姓名

PRIMARY key(empno)

) auto\_increment = 1000

-- 打卡信息表

create table signinfo(

id int auto\_increment, -- 主键

empno int default 0 not null, -- 工号

signdate date, -- 打卡日期

signin datetime,-- 签到时间

signout datetime, -- 签退时间

PRIMARY key(id)

)

1. 编写完整的JDBC操作工具类。

import java.io.IOException;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.util.Properties;

/\*\*

\* 1.节省资源 -- 单例

\* 2.开闭原则 -- 配置文件方式获取数据库信息

\* 3.获取链接 -- getConnection

\*

\* 常见错误：

\* ①配置文件必须在src下（classpath）

\* ②配置信息中不要有多余的符号，例如""

\* @author Administrator

\*/

public class DBUtils {

private static DBUtils db;

private String url\_;

private String user\_;

private String password\_;

private DBUtils() {

try {

// 加载配置文件信息

Properties p = new Properties();

p.load(this.getClass().getClassLoader().getResourceAsStream("db.properties"));

// getProperty通过key值获取配置文件中的value值（返回值）

url\_ = p.getProperty("url");

user\_ = p.getProperty("user");

password\_ = p.getProperty("password");

} catch (IOException e) {

e.printStackTrace();

}

}

public static DBUtils getInstance() {

if(db == null) {

db = new DBUtils();

}

return db;

}

/\*\*

\* 获取数据库连接的方法

\* @return 连接对象

\* @throws SQLException

\*/

public Connection getConnection() throws SQLException {

Connection conn = DriverManager.getConnection(url\_, user\_, password\_);

return conn;

}

}

1. 进入程序显示功能菜单，分别为录入员工信息、删除员工信息、修改员工信息，员工打卡，查看全部员工打卡信息，查看指定员工全部打卡信息，查看指定员工指定日期打卡信息，退出。

System.out.println("1.录入员工信息");

System.out.println("2.删除员工信息");

System.out.println("3.修改员工信息");

System.out.println("4.员工打卡");

System.out.println("5.查看全部员工打卡信息");

System.out.println("6.查看指定员工全部打卡信息");

System.out.println("7.查看指定员工指定日期打卡信息");

System.out.println("8.退出");

1. 完成员工录入、修改、删除功能

@Override

public void addEmp(String name) {

Connection conn = null;

PreparedStatement ps = null;

try {

conn = DBUtils.getInstance().getConnection();

conn.setAutoCommit(false);

ps = conn.prepareStatement("insert into emp(ename) values (?)");

ps.setString(1, name);

ps.executeUpdate();

conn.commit();

} catch (SQLException e) {

try {

conn.rollback();

} catch (SQLException e1) {

e1.printStackTrace();

}

e.printStackTrace();

} finally {

if(conn != null) {

try {

conn.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

if(ps != null) {

try {

ps.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

@Override

public int delEmp(int empno) {

Connection conn = null;

PreparedStatement ps = null;

int count = 0;

try {

conn = DBUtils.getInstance().getConnection();

conn.setAutoCommit(false);

ps = conn.prepareStatement("delete from emp where empno = ?");

ps.setInt(1, empno);

count = ps.executeUpdate();

conn.commit();

} catch (SQLException e) {

try {

conn.rollback();

} catch (SQLException e1) {

e1.printStackTrace();

}

e.printStackTrace();

} finally {

if(conn != null) {

try {

conn.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

if(ps != null) {

try {

ps.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

return count;

}

@Override

public int updateEmp(Emp e) {

Connection conn = null;

PreparedStatement ps = null;

int count = 0;

try {

conn = DBUtils.getInstance().getConnection();

conn.setAutoCommit(false);

ps = conn.prepareStatement("update emp set ename = ? where empno = ?");

// 从传入的对象当中获取编号和姓名

ps.setString(1, e.getEname());

ps.setInt(2, e.getEmpno());

count = ps.executeUpdate();

conn.commit();

} catch (SQLException e2) {

try {

conn.rollback();

} catch (SQLException e1) {

e1.printStackTrace();

}

e2.printStackTrace();

} finally {

if(conn != null) {

try {

conn.close();

} catch (SQLException e3) {

e3.printStackTrace();

}

}

if(ps != null) {

try {

ps.close();

} catch (SQLException e3) {

e3.printStackTrace();

}

}

}

return count;

}

1. 完成员工打卡功能

@Override

public int signIn(int empno) {

/\*

\* 1.查询empno工号员工是否存在（根据人数是0或1）

\* 2.1 不为0-- 打卡insert/签退update

\* 查询empno和当天打卡记录是否存在select count(id) idcounts from signinfo where empno = ? and date\_format(signin,'%Y-%m-%d') = ?

\* idcounts == 0 不存在 打卡insert

\* idcounts != 0 存在打卡记录update

\* 2.2 为0-- 提示不能打卡

\*

\* 常见错误：

\* ①函数或表达式的列必须起别名

\* ②yyyy-MM-dd 对应 %Y-%m-%d

\* ③date\_format(signin,'%Y-%m-%d') = ? 错写成 date\_format(signin,'%Y-%m-%d' = ?)

\*/

Connection conn = null;

PreparedStatement ps = null;

ResultSet rs = null;

int count = 0;

try {

conn = DBUtils.getInstance().getConnection();

// 1.查询工号为empno的人数，根据人数是0或1就能判断该员工是否可以打卡

ps = conn.prepareStatement("select count(empno) counts from emp where empno = ?");

ps.setInt(1, empno);

rs = ps.executeQuery();

if(rs.next()) {

count = rs.getInt("counts");

// 2.count不为0，代表有该工号员工，可以打卡/签退

if(count != 0) {

// 判断是打卡【insert】还是签退【update】

// 先查询signinfo表中是否有当天打卡记录

Date curDate = new Date();

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");

String strDate = sdf.format(curDate);

// 3.查询该工号员工，当天是否有打卡记录

ps = conn.prepareStatement("select count(id) idcounts from signinfo where empno = ? and date\_format(signin,'%Y-%m-%d') = ?");

ps.setInt(1, empno);

ps.setString(2, strDate);

rs = ps.executeQuery();

if(rs.next()) {

int idcounts = rs.getInt("idcounts");

conn.setAutoCommit(false);

if(idcounts == 0) {

// idcounts是0，表示今天没打过卡,本次操作就是正常打卡【insert】

ps = conn.prepareStatement("insert into signinfo(empno,signin) values (?,?)");

ps.setInt(1, empno);

Timestamp ts = new Timestamp(curDate.getTime());

ps.setTimestamp(2, ts);

System.out.println(empno+" 号员工签到成功");

}else {

// 非0，表示今天打过卡，本次操作就是签退【update】

ps = conn.prepareStatement("update signinfo set signout = ? where empno = ? and date\_format(signin,'%Y-%m-%d') = ?");

Timestamp ts = new Timestamp(curDate.getTime());

ps.setTimestamp(1, ts);

ps.setInt(2, empno);

ps.setString(3, strDate);

System.out.println(empno+" 号员工签退成功");

}

ps.executeUpdate();

conn.commit();

}

}else {

System.out.println("抱歉，您没有录入信息，不能打卡");

}

}

} catch (Exception e) {

try {

conn.rollback();

} catch (SQLException e1) {

e1.printStackTrace();

}

e.printStackTrace();

}finally {

if(conn != null) {

try {

conn.close();

} catch (SQLException e3) {

e3.printStackTrace();

}

}

if(ps != null) {

try {

ps.close();

} catch (SQLException e3) {

e3.printStackTrace();

}

}

if(rs != null) {

try {

rs.close();

} catch (SQLException e3) {

e3.printStackTrace();

}

}

}

return count;

}

1. 完成员工信息查询功能

@Override

public List<SignInfo> getSignInfo() {

Connection conn = null;

PreparedStatement ps = null;

ResultSet rs = null;

// 打卡名单

List<SignInfo> list = new ArrayList<SignInfo>();

try {

conn = DBUtils.getInstance().getConnection();

ps = conn.prepareStatement("select \* from signinfo");

rs = ps.executeQuery();

while(rs.next()) {

int id = rs.getInt("id");

int empno = rs.getInt("empno");

Date signin = rs.getTimestamp("signin");

Date signout = rs.getTimestamp("signout");

// 封装信息

SignInfo info = new SignInfo();

info.setId(id);

info.setEmpno(empno);

info.setSignin(signin);

info.setSignout(signout);

// 扔进list

list.add(info);

}

} catch (Exception e) {

e.printStackTrace();

} finally {

if(conn != null) {

try {

conn.close();

} catch (SQLException e3) {

e3.printStackTrace();

}

}

if(ps != null) {

try {

ps.close();

} catch (SQLException e3) {

e3.printStackTrace();

}

}

if(rs != null) {

try {

rs.close();

} catch (SQLException e3) {

e3.printStackTrace();

}

}

}

return list;

}

@Override

public List<NewSignInfo> getNewSignInfo() {

Connection conn = null;

PreparedStatement ps = null;

ResultSet rs = null;

// 打卡名单

List<NewSignInfo> list = new ArrayList<NewSignInfo>();

try {

conn = DBUtils.getInstance().getConnection();

ps = conn.prepareStatement("select e.empno,ename,signin,signout from signinfo s join emp e on e.empno = s.empno");

rs = ps.executeQuery();

while(rs.next()) {

int empno = rs.getInt("empno");

String ename = rs.getString("ename");

Date signin = rs.getTimestamp("signin");

Date signout = rs.getTimestamp("signout");

// 封装信息

NewSignInfo info = new NewSignInfo();

info.setEname(ename);

info.setEmpno(empno);

info.setSignIn(signin);

info.setSignOut(signout);

// 扔进list

list.add(info);

}

} catch (Exception e) {

e.printStackTrace();

} finally {

if(conn != null) {

try {

conn.close();

} catch (SQLException e3) {

e3.printStackTrace();

}

}

if(ps != null) {

try {

ps.close();

} catch (SQLException e3) {

e3.printStackTrace();

}

}

if(rs != null) {

try {

rs.close();

} catch (SQLException e3) {

e3.printStackTrace();

}

}

}

return list;

}

@Override

public List<Map<String, Object>> getNewSignInfoTwo() {

Connection conn = null;

PreparedStatement ps = null;

ResultSet rs = null;

// 打卡名单

List<Map<String, Object>> list = new ArrayList<>();

try {

conn = DBUtils.getInstance().getConnection();

ps = conn.prepareStatement("select e.empno,ename,signin,signout from signinfo s join emp e on e.empno = s.empno");

rs = ps.executeQuery();

while(rs.next()) {

int empno = rs.getInt("empno");

String ename = rs.getString("ename");

Date signin = rs.getTimestamp("signin");

Date signout = rs.getTimestamp("signout");

// 封装信息

Map<String, Object> map = new HashMap<String, Object>();

map.put("empno", empno);

map.put("ename", ename);

map.put("signin", signin);

map.put("signout", signout);

// 扔进list

list.add(map);

}

} catch (Exception e) {

e.printStackTrace();

} finally {

if(conn != null) {

try {

conn.close();

} catch (SQLException e3) {

e3.printStackTrace();

}

}

if(ps != null) {

try {

ps.close();

} catch (SQLException e3) {

e3.printStackTrace();

}

}

if(rs != null) {

try {

rs.close();

} catch (SQLException e3) {

e3.printStackTrace();

}

}

}

return list;

}