### JAVA第二阶段—DAY04-JAVA案例

1. 编写JDBC入门测试类类

* 需求

1、根据示例代码，手工获取jdbc连接，执行测试查询sql，打印输出结果集

* 答案

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.Statement;

public class MySQL\_QuickStart {

public void hello() throws Exception {

//1. 加载驱动

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

//2. 获得连接

Connection conn = DriverManager.getConnection

("jdbc:mysql://localhost:3306/dayxx", "root", "root");

//3. 编写sql

String sql = "select \* from category";

//4. 获得执行sql语句的对象

Statement st = conn.createStatement();

//5. 执行sql语句，并返回结果

ResultSet rs = st.executeQuery(sql);

//6. 处理结果

//判断是否还有下一条记录

while (rs.next()) {

//若有下一条,获取此条记录

Integer cid = rs.getInt("cid");

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

System.out.println(cid + " , " + cname);

}

//7. 释放资源

rs.close();

st.close();

conn.close();

}

public static void main(String[] args) throws Exception {

MySQL\_QuickStart t = new MySQL\_QuickStart();

t.hello();

}

}

1. 编写jdbc增删改查功能代码

* 需求

1、根据示例代码，分别编写jdbc-CURD代码案例

* 参考答案

public class Demo03 {

public static void main(String[] args) throws Exception {

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

Connection conn = DriverManager.getConnection("jdbc:mysql:///mysqltest", "root", "root");

System.out.println(conn);

// 从连接中拿到一个Statement对象

Statement stmt = conn.createStatement();

// 1.插入记录

String sql = "INSERT INTO category (cname) VALUES ('手机');";

int i = stmt.executeUpdate(sql);

System.out.println("影响的行数:" + i);

// 2.修改记录

sql = "UPDATE category SET cname='汽车' WHERE cid=4;";

i = stmt.executeUpdate(sql);

System.out.println("影响的行数:" + i);

// 3.删除记录

sql = "DELETE FROM category WHERE cid=1;";

i = stmt.executeUpdate(sql);

System.out.println("影响的行数:" + i);

// 释放资源

stmt.close();

conn.close();

}

}

1. 使用预处理语句方式，编写jdbc增删改查功能代码

* 需求

1. 根据示例代码，使用预处理语句的方式，分别编写jdbc-CURD代码案例

public class Demo03 {

public static void main(String[] args) throws Exception {

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

Connection conn = DriverManager.getConnection("jdbc:mysql:///mysqltest", "root", "root");

System.out.println(conn);

String sql = "INSERT INTO category (cname) VALUES (?)";

// 从连接中拿到一个Statement对象

PrepareStatement pstmt = conn.prepareStatement(sql);

// 1.插入记录

pstmt.setString(1, "小志");

int i = stmt.executeUpdate();

System.out.println("影响的行数:" + i);

pstmt.close();

// 2.修改记录

sql = "UPDATE category SET cname=? WHERE cid=?";

pstmt = conn.prepareStatement(sql);

pstmt.setString(1, "汽车");

pstmt.setInteger(2, 4);

i = stmt.executeUpdate();

System.out.println("影响的行数:" + i);

pstmt.close();

// 3.删除记录

sql = "DELETE FROM category WHERE cid=?";

pstmt = conn.prepareStatement(sql);

pstmt.setString(1, 5);

i = pstmt.executeUpdate();

System.out.println("影响的行数:" + i);

// 释放资源

pstmt.close();

conn.close();

}

}

1. 实现jdbc操作工具类

* 需求

根据示例代码，编写jdbc-操作工具类

public class JDBCUtil {

// 1.将固定字符串定义为常量

private static String DRIVER\_CLASS = "com.mysql.jdbc.Driver";

private static String URL = "jdbc:mysql:///dayxx";

private static String USER = "root";

private static String PASSWORD = "root";

// 2.在静态代码块中注册驱动(只注册一次)

// 当这个类加载到内存的时候就走这个静态代码块,再去触发Driver类中的静态代码块,主动注册

static {

/\* 可以将四个参数抽取为一个jdbc.properties文件，放置在src目录下，

然后通过Properties对象getProperty()方法获取每个值 \*/

InputStream is = JDBCUtil.class.getClassLoader()

.getResourceAsStream("jdbc.properties");

Properties pp = new Properties();

pp.load(is);

String driver\_class = pp.getProperty("DRIVER\_CLASS");

//如果dirver\_class确实配置了值，则

if(driver\_class != null && !driver\_class.trim().equals("")) {

DRIVER\_CLASS = driver\_class;

}

try {

Class.forName(DRIVER\_CLASS);

} catch (ClassNotFoundException e) {}

}

// 3.提供一个获取连接的方法static Connection getConneciton();

// 我们面向JDBC编程

public static Connection getConnection() throws SQLException {

Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);

/\* 可以将四个参数抽取为一个jdbc.properties文件，放置在src目录下，

然后通过Properties对象getProperty()方法获取每个值 \*/

InputStream is = JDBCUtil.class.getClassLoader()

.getResourceAsStream("jdbc.properties");

Properties pp = new Properties();

pp.load(is);

String driver\_name = properties.getProperty("DRIVER\_NAME");

if(driver\_name != null && !driver\_name.trim().equals("")) {

DRIVER\_NAME = driver\_name;

}

return conn;

}

// 5.重载关闭方法close(Connection conn, Statement stmt)

public static void close(Connection conn, Statement stmt) {

close(conn, stmt, null);

}

// 4.定义关闭资源的方法close(Connection conn, Statement stmt, ResultSet rs)

public static void close(Connection conn, Statement stmt, ResultSet rs) {

if (rs != null) {

try {

rs.close();

} catch (SQLException e) {}

}

if (stmt != null) {

try {

stmt.close();

} catch (SQLException e) {}

}

if (conn != null) {

try {

conn.close();

} catch (SQLException e) {}

}

}

}