**每日作业卷**

**原生jdbc**

# 关卡1

## 训练案例1

### 训练知识点

1. 客户端操作MySQL数据库的方式
2. JDBC的概念

### 训练描述

1. 客户端操作MySQL数据库的方式有哪些？

什么是JDBC?

### 操作步骤描述

JDBC

JDBC是一种用于执行SQL语句的Java API，可以为多种关系数据库提供统一访问，它由一组用[Java语言](https://baike.baidu.com/item/Java%E8%AF%AD%E8%A8%80)编写的类和接口组成。JDBC提供了一种基准，据此可以构建更高级的工具和接口，使数据库开发人员能够编写数据库应用程序，同时，JDBC也是个商标名。

## 训练案例2

### 训练知识点

1．JDBC连接数据库的参数

### 训练描述

使用JDBC连接数据库的四个参数分别是什么？

### 操作步骤描述

驱动名称 连接地址 用户名 密码

## 训练案例3

### 训练知识点

1. 数据库驱动的加载。
2. 获取数据库连接对象。

### 训练描述

加载MySQL数据库驱动并获得一个数据类库连接对象，将连接对象输出到控制台。

### 操作步骤描述

**package** com.java.test01;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**public** **class** DBTest {

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

String driver = "com.mysql.jdbc.Driver";

String url = "jdbc:mysql://localhost:3306/dbtest";

String username = "root";

String password = "123456";

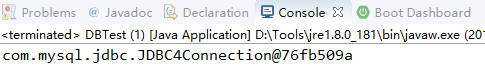
Class.*forName*(driver);

Connection connection = DriverManager.*getConnection*(url, username, password);

System.***out***.println(connection);

}

}



## 训练案例4

### 训练知识点

JDBC的开发步骤

### 训练描述

请描述JDBC的开发步骤?

### 操作步骤描述

（1） 加载JDBC驱动

（2） 建立并获取数据库连接

（3） 创建 JDBC Statements 对象

（4） 设置SQL语句的传入参数

（5） 执行SQL语句并获得查询结果

（6） 对查询结果进行转换处理并将处理结果返回

（7） 释放相关资源（关闭Connection，关闭Statement，关闭ResultSet）

## 训练案例5

### 训练知识点

1. 获得Statement对象
2. 执行DDL语句

### 训练描述

通过代码获得Statement对象并执行一条创表语句：创建一张学生表(id，姓名，性别)

### 操作步骤描述

1. 注册数据库驱动
2. 获得连接对象
3. 获得Statement对象
4. 定义要执行的创表语句
5. 调用Statement对象的方法执行SQL语句。
6. 释放资源。

**package** com.java.test01;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.Statement;

**public** **class** DBTest {

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

String driver = "com.mysql.jdbc.Driver";

String url = "jdbc:mysql://localhost:3306/dbtest";

String username = "root";

String password = "123456";

Class.*forName*(driver);

Connection connection = DriverManager.*getConnection*(url, username, password);

Statement statement = connection.createStatement();

String sql = "CREATE TABLE student(id INT PRIMARY KEY AUTO\_INCREMENT,name VARCHAR(20) ,gender VARCHAR(20))";

statement.execute(sql);

statement.close();

connection.close();

}

}



## 训练案例6

### 训练知识点

1.执行DML操作(insert)

### 训练描述

1．向学生表中添加1条记录，因为主键是自动增长，所以只需指定姓名和性别字段。

### 操作步骤描述

1. 注册数据库驱动
2. 获得连接对象
3. 获得Statement对象
4. 准备SQL语句
5. 调用Statement对象的方法执行SQL语句。
6. 释放资源

**package** com.java.test01;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.Statement;

**public** **class** DBTest {

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

String driver = "com.mysql.jdbc.Driver";

String url = "jdbc:mysql://localhost:3306/dbtest";

String username = "root";

String password = "123456";

Class.*forName*(driver);

Connection connection = DriverManager.*getConnection*(url, username, password);

Statement statement = connection.createStatement();

String sql = "INSERT INTO student(NAME,gender) VALUE ('11','11')";

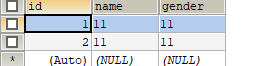
statement.execute(sql);

statement.close();

connection.close();

}

}



## 训练案例7

### 训练知识点

1.执行更新操作(update)

### 训练描述

将id为2的用户，姓名更新为"猪八戒"

### 操作步骤描述

**package** com.java.test01;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.Statement;

**public** **class** DBTest {

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

String driver = "com.mysql.jdbc.Driver";

String url = "jdbc:mysql://localhost:3306/dbtest";

String username = "root";

String password = "123456";

Class.*forName*(driver);

Connection connection = DriverManager.*getConnection*(url, username, password);

Statement statement = connection.createStatement();

String sql = "update student set name = '猪八戒' where id = 2 ";

statement.execute(sql);

statement.close();

connection.close();

}

}



## 训练案例8

### 训练知识点

1. ResultSet接口的使用
2. 执行DQL操作

### 训练描述

1.确保数据库中有2条以上的记录，查询所有的学员信息，输出到控制台

### 操作步骤描述

1. 注册数据库驱动
2. 获得数据库连接对象
3. 创建Statement对象
4. 准备SQL查询语句
5. 执行SQL，返回结果
6. 使用while循环遍历

**package** com.java.test01;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.ResultSet;

**import** java.sql.Statement;

**public** **class** DBTest {

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

String driver = "com.mysql.jdbc.Driver";

String url = "jdbc:mysql://localhost:3306/dbtest";

String username = "root";

String password = "123456";

Class.*forName*(driver);

Connection connection = DriverManager.*getConnection*(url, username, password);

Statement statement = connection.createStatement();

String sql = "select \* from student";

ResultSet rs = statement.executeQuery(sql);

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

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

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

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

System.***out***.println(id + " " + name + " " + gender);

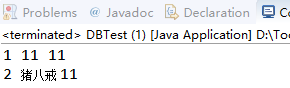
}

statement.close();

connection.close();

}

}



## 训练案例9

### 9.1 训练知识点

1．数据库工具类

### 9.2 训练描述

定义一个数据库工具类JdbcUtil，用来简化数据库操作出现的重复代码。

创建类JdbcUtil包含3个方法：

1) 可以把几个字符串定义成常量

2) public static Connection getConnection() 得到数据库的连接

3) 在静态代码块中注册驱动，只需注册一次即可。无需放在getConnection()方法中

4) public static void close(Connection conn,Statement stmt,ResultSet rs) 关闭所有打开的资源

5)public static void close(Connection conn,Statement stmt) 关闭没有结果集的资源，可以调用上面的方法。

### 操作步骤描述

**package** com.java.util;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.sql.Statement;

**public** **class** JdbcUtil {

**private** **static** String *driver* = "com.mysql.jdbc.Driver";

**private** **static** String *url* = "jdbc:mysql://localhost:3306/dbtest";

**private** **static** String *username* = "root";

**private** **static** String *password* = "123456";

**static** {

**try** {

Class.*forName*(*driver*);

} **catch** (ClassNotFoundException e) {

e.printStackTrace();

}

}

**public** **static** Connection getConnection() **throws** Exception {

Connection connection = DriverManager.*getConnection*(*url*, *username*, *password*);

**return** connection;

}

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

**if** (rs != **null**) {

**try** {

rs.close();

} **catch** (SQLException e) {

e.printStackTrace();

}

}

**if** (stmt != **null**) {

**try** {

stmt.close();

} **catch** (SQLException e) {

e.printStackTrace();

}

}

**if** (conn != **null**) {

**try** {

conn.close();

} **catch** (SQLException e) {

e.printStackTrace();

}

}

}

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

**if** (stmt != **null**) {

**try** {

stmt.close();

} **catch** (SQLException e) {

e.printStackTrace();

}

}

**if** (conn != **null**) {

**try** {

conn.close();

} **catch** (SQLException e) {

e.printStackTrace();

}

}

}

}

## 10. 训练案例10

### 10.1 训练知识点

1． JDBC 预处理对象--增删改查

### 10.2 训练描述

使用PreparedStatement对象完成数据库的增删改查

1.添加一个学生到学生表

2.删除id=1的学生信息

3.修改id=2学生的姓名为”jack”

4.查询所有的学生信息

### 10.3 操作步骤描述

**package** com.java.test01;

**import** java.sql.Connection;

**import** java.sql.PreparedStatement;

**import** java.sql.ResultSet;

**import** org.junit.Test;

**import** com.java.util.JdbcUtil;

**public** **class** DBTest02 {

Connection connection = **null**;

PreparedStatement pstm = **null**;

ResultSet rs = **null**;

@Test

**public** **void** add() {

**try** {

connection = JdbcUtil.*getConnection*();

String sql = "INSERT INTO student(NAME,gender) VALUE (?,?)";

pstm = connection.prepareStatement(sql);

pstm.setString(1, "44");

pstm.setString(2, "44");

**int** r = pstm.executeUpdate();

System.***out***.println(r);

} **catch** (Exception e) {

e.printStackTrace();

} **finally** {

JdbcUtil.*close*(connection, pstm);

}

}

@Test

**public** **void** delete() {

**try** {

connection = JdbcUtil.*getConnection*();

String sql = "delete from student where id = 1";

pstm = connection.prepareStatement(sql);

**int** r = pstm.executeUpdate();

System.***out***.println(r);

} **catch** (Exception e) {

e.printStackTrace();

} **finally** {

JdbcUtil.*close*(connection, pstm);

}

}

@Test

**public** **void** update() {

**try** {

connection = JdbcUtil.*getConnection*();

String sql = "update student set name = 'jack' where id = 2";

pstm = connection.prepareStatement(sql);

**int** r = pstm.executeUpdate();

System.***out***.println(r);

} **catch** (Exception e) {

e.printStackTrace();

} **finally** {

JdbcUtil.*close*(connection, pstm);

}

}

@Test

**public** **void** search() {

**try** {

connection = JdbcUtil.*getConnection*();

String sql = "select \* from student";

pstm = connection.prepareStatement(sql);

rs = pstm.executeQuery();

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

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

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

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

System.***out***.println(id + " " + name + " " + gender);

}

} **catch** (Exception e) {

e.printStackTrace();

} **finally** {

JdbcUtil.*close*(connection, pstm, rs);

}

}

}

# 关卡2

## 训练案例1

### 训练知识点

1. 使用JDBC操作数据库数据
2. statement接口的使用

### 训练描述

模拟用户登录：需求如下

1.创建一张用户表(id,name,password)

2.添加几条用户记录

3.从键盘上输入用户和密码，拼接SQL语句

4.使用Statement执行SQL语句。

以上所有的操作要求使用JDBC相关API完成。

### 操作步骤描述

1. 注册数据库驱动
2. 获得数据库连接对象
3. 执行DDL创表语句
4. 执行DML插入用户数据语句
5. 执行DQL查询语句
6. 关闭相关资源。

**package** com.java.test02;

**import** java.sql.Connection;

**import** java.sql.PreparedStatement;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.sql.Statement;

**import** java.util.Scanner;

**import** org.junit.Test;

**import** com.java.util.JdbcUtil;

**public** **class** Test02 {

Connection connection = **null**;

PreparedStatement pstm = **null**;

ResultSet rs = **null**;

@Test

**public** **void** createTable() {

**try** {

connection = JdbcUtil.*getConnection*();

String sql = "CREATE TABLE user(id INT PRIMARY KEY AUTO\_INCREMENT,name VARCHAR(20) ,password VARCHAR(20))";

Statement stmt = connection.createStatement();

stmt.execute(sql);

stmt.close();

connection.close();

} **catch** (Exception e) {

e.printStackTrace();

}

}

@Test

**public** **void** add() {

**try** {

connection = JdbcUtil.*getConnection*();

String sql = "INSERT INTO user(name,password) VALUE ('www','123')";

Statement stmt = connection.createStatement();

stmt.execute(sql);

stmt.close();

connection.close();

} **catch** (Exception e) {

e.printStackTrace();

}

}

@Test

**public** **void** login() {

**try** {

connection = JdbcUtil.*getConnection*();

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.println("输入用户名：");

String username = scanner.nextLine();

System.***out***.println("输入密码：");

String password = scanner.nextLine();

String sql = "SELECT \* FROM USER WHERE NAME = '" + username + "' AND PASSWORD = '" + password + "'";

Statement stmt = connection.createStatement();

rs = stmt.executeQuery(sql);

**if** (rs.first()) {

System.***out***.println("登录成功");

} **else** {

System.***out***.println("登录失败");

}

stmt.close();

connection.close();

rs.close();

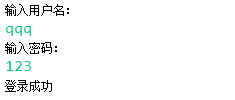
} **catch** (Exception e) {

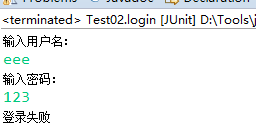
e.printStackTrace();

}

}

}





## 训练案例2

### 训练知识点

1. PreparedStatement接口的使用

### 训练描述

1.编写方法，通过ID查询指定学号的学生信息，并封装成Student对象返回。

### 操作步骤描述

package com.java.test03;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.Statement;

import org.junit.Test;

import com.java.util.JdbcUtil;

public class Test03 {

Connection connection = null;

PreparedStatement pstm = null;

ResultSet rs = null;

@Test

public void search() {

try {

Student student = new Student();

connection = JdbcUtil.getConnection();

String sql = "select \* from student where id = '2'";

PreparedStatement pstm = connection.prepareStatement(sql);

rs = pstm.executeQuery();

while (rs.next()) {

student.setId(rs.getInt("id"));

student.setName(rs.getString("name"));

student.setGender(rs.getString("gender"));

}

System.out.println(student);

rs.close();

pstm.close();

connection.close();

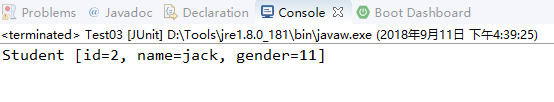
} catch (Exception e) {

e.printStackTrace();

}

}

}



## 训练案例3

### 训练知识点

1. PreparedStatement接口的使用

### 训练描述

1. 编写方法，通过ID删除指定学号的学生，并且返回删除的行数。

### 操作步骤描述

@Test

**public** **void** delete() {

**try** {

connection = JdbcUtil.*getConnection*();

String sql = "delete from student where id = 2";

PreparedStatement pstm = connection.prepareStatement(sql);

**int** r = pstm.executeUpdate();

System.***out***.println("删除了" + r + "行");

pstm.close();

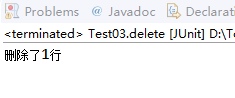
connection.close();

} **catch** (Exception e) {

e.printStackTrace();

}

}



## 训练案例4

### 训练知识点

1.PreparedStatement接口的使用

2.ResultSet接口的使用

### 训练描述

1.编写方法，查询学生表所有的学生，封装成List<Student>返回

### 操作步骤描述

@Test

**public** **void** search2() {

**try** {

ArrayList<Student> arrayList = **new** ArrayList<Student>();

connection = JdbcUtil.*getConnection*();

String sql = "select \* from student ";

PreparedStatement pstm = connection.prepareStatement(sql);

rs = pstm.executeQuery();

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

Student student = **new** Student();

student.setId(rs.getInt("id"));

student.setName(rs.getString("name"));

student.setGender(rs.getString("gender"));

arrayList.add(student);

}

**for** (Student student : arrayList) {

System.***out***.println(student);

}

rs.close();

pstm.close();

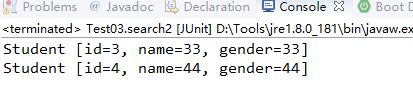
connection.close();

} **catch** (Exception e) {

e.printStackTrace();

}

}



# 关卡3

## 训练案例1

### 训练知识点

1. SQL语句的编写
2. JDBC操作MySQL数据库常用API

### 训练描述

1.在数据库创建部门表和员工表，包含字段如下：

部门表包含字段有：部门编号，部门名称(唯一且不能为空)

员工表包含字段有：员工编号，员工姓名(唯一且不能为空)，员工性别，员工职位，员工工资，入职日期。部门编号(外键)

2.先添加多个部门数据，再添加多条员工数据

3.编写方法接收一个员工编号和工资两个参数，方法内将指定编号的员工工资修改为新的工资。

4.编写方法查询指定职位所有员工的信息，返回List<Employee>集合。

5.编写方法查询指定姓名的员工信息，返回Employee对象。

6.编写方法根据员工姓名删除指定的员工信息。

7.编写方法查询所有姓张员工的工资并输出在控制台，输出格式如下：

张三=10000

张飞=20000

…………………..

8.编写方法接收一个工资参数，方法内查询工资大于等于传入的工资的员工，返回符合条件所有员工信息List<Employee>集合。

9.编写方法查询指定部门的所有员工信息，返回List<Employee>集合

**package** com.java.test04;

**import** java.sql.Connection;

**import** java.sql.Date;

**import** java.sql.PreparedStatement;

**import** java.sql.ResultSet;

**import** java.util.ArrayList;

**import** com.java.util.JdbcUtil;

**public** **class** Test04 {

Connection connection = **null**;

PreparedStatement pstm = **null**;

ResultSet rs = **null**;

**public** **void** function1(**int** id, **double** salary) {

**try** {

connection = JdbcUtil.*getConnection*();

String sql = "update employee set e\_salary = " + salary + " where e\_id = " + id + " ";

pstm = connection.prepareStatement(sql);

**int** r = pstm.executeUpdate();

pstm.close();

connection.close();

} **catch** (Exception e) {

e.printStackTrace();

}

}

**public** **void** function2(String job) {

**try** {

connection = JdbcUtil.*getConnection*();

String sql = "SELECT e.e\_id,e.e\_name,e.e\_gender,e.e\_job,e.e\_salary,e.e\_date,d.d\_name FROM employee e "

+ "INNER JOIN department d ON e.d\_id=d.d\_id " + "WHERE e\_job = '" + job + "'";

pstm = connection.prepareStatement(sql);

rs = pstm.executeQuery();

ArrayList<Employee> arrayList = **new** ArrayList<>();

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

**int** e\_id = rs.getInt("e\_id");

String e\_name = rs.getString("e\_name");

String e\_gender = rs.getString("e\_gender");

String e\_job = rs.getString("e\_job");

**double** e\_salary = rs.getDouble("e\_salary");

Date e\_date = rs.getDate("e\_date");

String d\_name = rs.getString("d\_name");

Employee employee = **new** Employee(e\_id, e\_name, e\_gender, e\_job, e\_salary, e\_date, d\_name);

arrayList.add(employee);

}

**for** (Employee employee : arrayList) {

System.***out***.println(employee.toString());

}

rs.close();

pstm.close();

connection.close();

} **catch** (Exception e) {

e.printStackTrace();

}

}

**public** **void** function3(String name) {

**try** {

connection = JdbcUtil.*getConnection*();

String sql = "SELECT e.e\_id,e.e\_name,e.e\_gender,e.e\_job,e.e\_salary,e.e\_date,d.d\_name FROM employee e "

+ "INNER JOIN department d ON e.d\_id=d.d\_id " + "WHERE e\_name = '" + name + "'";

pstm = connection.prepareStatement(sql);

rs = pstm.executeQuery();

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

**int** e\_id = rs.getInt("e\_id");

String e\_name = rs.getString("e\_name");

String e\_gender = rs.getString("e\_gender");

String e\_job = rs.getString("e\_job");

**double** e\_salary = rs.getDouble("e\_salary");

Date e\_date = rs.getDate("e\_date");

String d\_name = rs.getString("d\_name");

Employee employee = **new** Employee(e\_id, e\_name, e\_gender, e\_job, e\_salary, e\_date, d\_name);

System.***out***.println(employee.toString());

}

rs.close();

pstm.close();

connection.close();

} **catch** (Exception e) {

e.printStackTrace();

}

}

**public** **void** function4(String name) {

**try** {

connection = JdbcUtil.*getConnection*();

String sql = "delete from employee where e\_name = '" + name + "'";

pstm = connection.prepareStatement(sql);

**int** r = pstm.executeUpdate();

pstm.close();

connection.close();

} **catch** (Exception e) {

e.printStackTrace();

}

}

**public** **void** function5() {

**try** {

connection = JdbcUtil.*getConnection*();

String sql = "SELECT e.e\_name,e.e\_salary FROM employee e WHERE e.e\_name LIKE '张%'";

pstm = connection.prepareStatement(sql);

rs = pstm.executeQuery();

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

String e\_name = rs.getString("e\_name");

**double** e\_salary = rs.getDouble("e\_salary");

System.***out***.println(e\_name + "=" + e\_salary);

}

rs.close();

pstm.close();

connection.close();

} **catch** (Exception e) {

e.printStackTrace();

}

}

**public** **void** function6(**double** salary) {

**try** {

connection = JdbcUtil.*getConnection*();

String sql = "SELECT e.e\_id,e.e\_name,e.e\_gender,e.e\_job,e.e\_salary,e.e\_date,d.d\_name FROM employee e "

+ "INNER JOIN department d ON e.d\_id=d.d\_id WHERE e\_salary >= '" + salary + "'";

pstm = connection.prepareStatement(sql);

rs = pstm.executeQuery();

ArrayList<Employee> arrayList = **new** ArrayList<>();

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

**int** e\_id = rs.getInt("e\_id");

String e\_name = rs.getString("e\_name");

String e\_gender = rs.getString("e\_gender");

String e\_job = rs.getString("e\_job");

**double** e\_salary = rs.getDouble("e\_salary");

Date e\_date = rs.getDate("e\_date");

String d\_name = rs.getString("d\_name");

Employee employee = **new** Employee(e\_id, e\_name, e\_gender, e\_job, e\_salary, e\_date, d\_name);

arrayList.add(employee);

}

**for** (Employee employee : arrayList) {

System.***out***.println(employee.toString());

}

rs.close();

pstm.close();

connection.close();

} **catch** (Exception e) {

e.printStackTrace();

}

}

**public** **void** function7(String departmrnt) {

**try** {

connection = JdbcUtil.*getConnection*();

String sql = "SELECT e.e\_id,e.e\_name,e.e\_gender,e.e\_job,e.e\_salary,e.e\_date,d.d\_name FROM employee e "

+ "INNER JOIN department d ON e.d\_id=d.d\_id WHERE d.d\_name = '" + departmrnt + "'";

pstm = connection.prepareStatement(sql);

rs = pstm.executeQuery();

ArrayList<Employee> arrayList = **new** ArrayList<>();

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

**int** e\_id = rs.getInt("e\_id");

String e\_name = rs.getString("e\_name");

String e\_gender = rs.getString("e\_gender");

String e\_job = rs.getString("e\_job");

**double** e\_salary = rs.getDouble("e\_salary");

Date e\_date = rs.getDate("e\_date");

String d\_name = rs.getString("d\_name");

Employee employee = **new** Employee(e\_id, e\_name, e\_gender, e\_job, e\_salary, e\_date, d\_name);

arrayList.add(employee);

}

**for** (Employee employee : arrayList) {

System.***out***.println(employee.toString());

}

rs.close();

pstm.close();

connection.close();

} **catch** (Exception e) {

e.printStackTrace();

}

}

**public** **static** **void** main(String[] args) {

Test04 t = **new** Test04();

// t.function1(1, 1500);

// t.function2("清洁工");

// t.function3("李四");

// t.function4("王五");'

// t.function5();

// t.function6(1300);

t.function7("后勤部");

}

}