**每日作业卷答案**

**原生jdbc**

# 关卡1

## 训练案例1

### 训练描述

客户端操作MySQL数据库的方式有哪些？

什么是JDBC?

### 操作步骤答案

客户端操作MySQL数据库的方式有哪些

\* 使用第三方客户端来访问MySQL

\* SQLyog、Navicat、SQLWave、MyDB Studio、EMS SQL Manager for MySQLD等

\* 使用MySQL自带的命令行方式

\* 通过Java程序来访问MySQL数据库。

什么是JDBC

\* JDBC全称：Java DataBase Connectivity -- Java的数据库连接技术。

\* 它由一组使用Java语言编写的类与接口(如何实现接口由各大数据库厂商来实现)组成，为各种关系型数据库提供了统一访问方式。

## 训练案例2

### 训练描述

使用JDBC连接数据库的四个参数分别是什么？

### 操作步骤答案

用户名：root

密码：root

连接字符串：jdbc:mysql://localhost:3306/数据库

数据库驱动类：com.mysql.jdbc.Driver

## 训练案例3

### 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。

### 操作步骤答案

|  |
| --- |
| **public** **class** Test01\_03 {  **public** **static** **void** main(String[] args) {  // 数据库地址  String url = "jdbc:mysql://localhost:3306/mydb";  // 用户名  String user = "root";  // 密码  String password = "123456";  // 注册数据库驱动  Class.*forName*("com.mysql.jdbc.Driver");  // 连接数据库并获得连接对象  Connection conn = DriverManager.*getConnection*(url, user, password);  System.***out***.println(conn);  }  } |

## 训练案例4

### 训练描述

请描述JDBC的开发步骤?

### 操作步骤答案

|  |
| --- |
| 1. 注册并加载数据库驱动 Class.forName()  2. 连接数据库，获取连接对象 Connection conn = DriverManager.getConnection();  3. 创建Statement对象 Statement stmt = conn.createStatement();  4. 使用Statement对象发送SQL  5. 处理结果  6. 关闭连接，后开的先关。 |

## 训练案例5

### 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)

### 操作步骤答案

|  |
| --- |
| **public** **static** **void** main(String[] args) **throws** Exception {  // 连接数据库参数  String url = "jdbc:mysql://localhost:3306/mydb";  String user = "root";  String password = "123456";  // 注册数据库驱动  Class.*forName*("com.mysql.jdbc.Driver");  // 获得数据库连接对象  Connection conn = DriverManager.*getConnection*(url, user, password);  // 获得statement对象  Statement stmt = conn.createStatement();  // 创表SQL语句  String sql = "create table stu("  + " id int primary key auto\_increment,"  + " name varchar(20),"  + " gender varchar(2));";  // 执行创表SQL语句  **int** row = stmt.executeUpdate(sql);        // ddl的操作不影响行数      System.***out***.println(row);    System.***out***.println("创建学生表成功");  }  } |

## 训练案例6

### 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。

### 操作步骤答案

|  |
| --- |
| **public** **static** **void** main(String[] args) {  // 连接对象  Connection conn = **null**;  // SQL语句对象  Statement state = **null**;  **try** {  // 数据库地址  String url = "jdbc:mysql:///mydb";  // 用户名  String user = "root";  // 密码  String password = "123456";  // 注册数据库驱动  Class.*forName*("com.mysql.jdbc.Driver");  // 连接数据库并获得连接对象  conn = DriverManager.*getConnection*(url, user, password);  // 获得SQL语句对象  state = conn.createStatement();  // 要执行的sql语句  String sql = "insert into stu(name,gender) values('小黑','男')";  // 执行sql语句  **int** row = state.executeUpdate(sql);  System.***out***.println(row);  } **catch** (ClassNotFoundException | SQLException e) {  e.printStackTrace();  } **finally** {  // 释放资源  **try** {  **if**(state != **null**)  state.close();  } **catch** (SQLException e) {  e.printStackTrace();  }    **try** {  **if**(conn != **null**)  conn.close();  } **catch** (SQLException e) {  e.printStackTrace();  }  }  } |

## 训练案例7

### 训练描述

1.将id为2的用户，姓名更新为"猪八戒"

### 操作步骤答案

|  |
| --- |
| **public** **static** **void** main(String[] args) {  // 连接对象  Connection conn = **null**;  // SQL语句对象  Statement state = **null**;  **try** {  String name = "猪八戒";  **int** id = 2;  // 数据库地址  String url = "jdbc:mysql:///mydb";  // 用户名  String user = "root";  // 密码  String password = "123456";  // 注册数据库驱动  Class.*forName*("com.mysql.jdbc.Driver");  // 连接数据库并获得连接对象  conn = DriverManager.*getConnection*(url, user, password);  // 获得SQL语句对象  state = conn.createStatement();  // 要执行的sql语句  String sql = "update stu set name='"+name+"' where id="+id+";";  // 执行sql语句  **int** row = state.executeUpdate(sql);  System.***out***.println(row);  } **catch** (ClassNotFoundException | SQLException e) {  e.printStackTrace();  } **finally** {  // 释放资源  **try** {  **if**(state != **null**)  state.close();  } **catch** (SQLException e) {  e.printStackTrace();  }    **try** {  **if**(conn != **null**)  conn.close();  } **catch** (SQLException e) {  e.printStackTrace();  }  }  } |

## 训练案例8

### 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

### 操作步骤答案

|  |
| --- |
| **public** **static** **void** main(String[] args) **throws** Exception {  // 连接对象  Connection conn = **null**;  // SQL语句对象  Statement state = **null**;  // 查询结果集  ResultSet resultSet = **null**;  **try** {  // 数据库地址  String url = "jdbc:mysql:///mydb";  // 用户名  String user = "root";  // 密码  String password = "123456";  // 注册数据库驱动  Class.*forName*("com.mysql.jdbc.Driver");  // 连接数据库并获得连接对象  conn = DriverManager.*getConnection*(url, user, password);  // 获得SQL语句对象  state = conn.createStatement();  // 要执行的sql语句  String sql = "select \* from stu";  // 执行sql语句  resultSet = state.executeQuery(sql);  **while**(resultSet.next()) {  // 根据列号获得字段值  **int** id = resultSet.getInt(1);  String name = resultSet.getString(2);  String gender = resultSet.getString(3);  System.***out***.println("学号："+id + " 姓名：" + name + "  性别：" + gender);  System.***out***.println("-----------------------");  // 根据列名获得字段值  id = resultSet.getInt("id");  name = resultSet.getString("name");  name = resultSet.getString("gender");  System.***out***.println("学号："+id + " 姓名：" + name + "  性别：" + gender);  }  } **catch** (ClassNotFoundException | SQLException e) {  e.printStackTrace();  } **finally** {  // 释放资源  **try** {  **if**(resultSet != **null**)  resultSet.close();  } **catch** (SQLException e1) {  e1.printStackTrace();  }  **try** {  **if**(state != **null**)  state.close();  } **catch** (SQLException e) {  e.printStackTrace();  }  **try** {  **if**(conn != **null**)  conn.close();  } **catch** (SQLException e) {  e.printStackTrace();  }  }  } |

## 训练案例9

### 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。

### 操作步骤答案

|  |
| --- |
| **public** **class** JDBCUtil {  // 连接数据库的URL  **private** **static** **final** String ***URL*** = "jdbc:mysql://localhost:3306/mydb";    // 数据库用户名  **private** **static** **final** String ***USER*** = "root";    // 数据库密码  **private** **static** **final** String ***PASSWORD*** = "123456";    // 驱动程序类  **private** **static** **final** String ***DRIVER\_CLASS*** = "com.mysql.jdbc.Driver";    /\*\*    \* 只注册驱动程序一次就可以(静态代码块中)    \*/  **static** {  **try** {        // 注册驱动        Class.*forName*(***DRIVER\_CLASS***);      } **catch** (ClassNotFoundException e) {        e.printStackTrace();      }    }    /\*\*    \* 获取连接方法    \*    \* **@return**    \*/  **public** **static** Connection getConnection() {  **try** {  // 获取连接  Connection conn = DriverManager.*getConnection*(***URL***, ***USER***, ***PASSWORD***);  **return** conn;  } **catch** (SQLException e) {  e.printStackTrace();  }  **return** **null**;  }  /\*\*  \* 关闭资源方法（先关闭ResultSet，再关闭Statement，最后关闭Connection）  \*/  **public** **static** **void** close(Connection conn, Statement stmt, ResultSet rs) {  **try** {  **if** (rs != **null**) {        rs.close();      }    } **catch** (SQLException e) {      e.printStackTrace();    }  **try** {  **if** (stmt != **null**) {        stmt.close();      }    } **catch** (SQLException e) {      e.printStackTrace();    }  **try** {  **if** (conn != **null**) {        conn.close();      }    } **catch** (SQLException e) {      e.printStackTrace();    }  }    /\*\*    \* 关闭只有2个对象的方法    \*/  **public** **static** **void** close(Connection conn, Statement stmt) {  *close*(conn, stmt, **null**);    }  } |

## 训练案例10

### 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

### 操作步骤答案

|  |
| --- |
| **public** **static** **void** main(String[] args) **throws** Exception{  //addStudent();  //deleteStudent();  //updateStudent();  *findAllStudent*();  }    // 1.添加一个学生到学生表  **public** **static** **void** addStudent() **throws** Exception{  // 获得连接对象  Connection conn = JdbcUtil.*getConnection*();  // 创建PreparedStatement对象  PreparedStatement ps = conn.prepareStatement("insert into stu(name,gender) values('jack','男')");  // 执行添加操作  **int** row = ps.executeUpdate();  System.***out***.println(row);  // 关闭资源  JdbcUtil.*close*(conn, ps);  }    // 2.删除id=1的学生信息  **public** **static** **void** deleteStudent() **throws** Exception{  // 获得连接对象  Connection conn = JdbcUtil.*getConnection*();  // 创建PreparedStatement对象  PreparedStatement ps = conn.prepareStatement("delete from stu where id = ? ");  // 设置参数  ps.setInt(1, 1);  // 执行添加操作  **int** row = ps.executeUpdate();  System.***out***.println(row);  // 关闭资源  JdbcUtil.*close*(conn, ps);  }    // 3.修改id=2学生的姓名为”jack”  **public** **static** **void** updateStudent() **throws** Exception{  // 获得连接对象  Connection conn = JdbcUtil.*getConnection*();  // 创建PreparedStatement对象  PreparedStatement ps = conn.prepareStatement("update stu set name = ? where id = ?");  // 设置参数  ps.setString(1, "jack");  ps.setInt(2, 2);  // 执行添加操作  **int** row = ps.executeUpdate();  System.***out***.println(row);  // 关闭资源  JdbcUtil.*close*(conn, ps);  }    // 4.查询所有的学生信息  **public** **static** **void** findAllStudent() **throws** Exception{  // 获得连接对象  Connection conn = JdbcUtil.*getConnection*();  // 创建PreparedStatement对象  PreparedStatement ps = conn.prepareStatement("select \* from stu;");  // 执行查询操作  ResultSet rs = ps.executeQuery();  // 循环获得每一条查询记录  **while**(rs.next()) {  // 获得学号  **int** id = rs.getInt("id");  // 获得姓名  String name = rs.getString("name");  // 获得性别  String gender = rs.getString("gender");  System.***out***.println("学号：" + id + ",姓名：" + name + ",性别：" + gender);  }  // 关闭资源  JdbcUtil.*close*(conn, ps,rs);  } |

# 关卡2

## 训练案例1

### 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。

### 操作步骤答案

|  |
| --- |
| **public** **static** **void** main(String[] args) **throws** Exception {  // 创建用户表  //createUserTable();  // 插入用户数据  // insertUserData();  // 用户登录  *login*();  }    /\*  \* 用户登录  \*/  **private** **static** **void** login() {  //1.要求用户输入用户名和密码  System.***out***.println("请输入用户名：");  Scanner sc = **new** Scanner(System.***in***);  String name = sc.nextLine();  System.***out***.println("请输入密码：");  String password = sc.nextLine();  //调用登录的方法  *login*(name, password);  }    /\*\*  \* 通过访问数据库来判断是否登录成功  \* **@param** name  \* **@param** password  \*/  **public** **static** **void** login(String name, String password) {  Connection conn = **null**;  Statement stmt = **null**;  ResultSet rs = **null**;  String sql = "select \* from user where name='" + name + "' and password='" + password + "'";  System.***out***.println(sql);  **try** {  //1.得到连接  conn = JdbcUtil.*getConnection*();  //2. 创建语句对象  stmt = conn.createStatement();  //3. 查询数据库得到结果集  rs = stmt.executeQuery(sql);  //4. 查看是否有结果集  **if** (rs.next()) {  //通过列名读取  System.***out***.println("欢迎您！" + rs.getString("name"));  } **else** {  System.***out***.println("用户名或密码不正确");  }  } **catch** (SQLException e) {  e.printStackTrace();  } **finally** {  JdbcUtil.*close*(conn, stmt, rs);  }  }    /\*  \* 插入用户数据  \*/  **private** **static** **void** insertUserData() {  // 连接对象  Connection conn = **null**;  // SQL语句对象  Statement state = **null**;  **try** {  // 获得连接对象  conn = JdbcUtil.*getConnection*();  // 创建SQL语句对象  state = conn.createStatement();  // 执行SQL语句  **int** row = state.executeUpdate("insert into user(name,password) values"  + "('张三','123456'),"  + "('李四','123');");  System.***out***.println("插入了" +row+"条记录");  } **catch** (SQLException e) {  e.printStackTrace();  }**finally** {  // 关闭资源  JdbcUtil.*close*(conn, state);  }  }  /\*  \* 创建用户表  \*/  **private** **static** **void** createUserTable() {  // 连接对象  Connection conn = **null**;  // SQL语句对象  Statement state = **null**;  **try** {  // 获得连接对象  conn = JdbcUtil.*getConnection*();  // 创建SQL语句对象  state = conn.createStatement();  // 执行创表语句  **int** row = state.executeUpdate("create table user(id int primary key auto\_increment,"  + "name varchar(20) unique,"  + "password varchar(20))");  System.***out***.println(row);  } **catch** (SQLException e) {  e.printStackTrace();  }**finally** {  // 关闭资源  JdbcUtil.*close*(conn, state);  }  } |

## 训练案例2

### 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。

### 操作步骤答案

|  |
| --- |
| **public** **static** **void** main(String[] args) {  Student stu = *findStudentById*(3);  System.***out***.println(stu);  }    // 根据id查询学生信息  **public** **static** Student findStudentById(**int** id) {  String sql = "select id,name,gender from stu where id = ? ";  // 得到连接对象  Connection conn = JdbcUtil.*getConnection*();  // 创建PreparedStatement  PreparedStatement ps  = **null**;  ResultSet rs = **null**;  // 创建学生对象  Student s = **new** Student();  **try** {  ps = conn.prepareStatement(sql);  // 设置参数并执行  ps.setInt(1, id);  rs = ps.executeQuery();  // 指针下移  rs.next();  // 封装数据  s.setId(rs.getInt("id"));  s.setName(rs.getString("name"));  s.setGender(rs.getString("gender"));  } **catch** (SQLException e) {  e.printStackTrace();  }  **finally** {  JdbcUtil.*close*(conn, ps, rs);  }  **return** s;  } |

## 训练案例3

### 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。

### 操作步骤答案

|  |
| --- |
| **public** **static** **void** main(String[] args) {  System.***out***.println(*deleteStuById*(3));  }    **public** **static** **int** deleteStuById(**int** id) {  String sql = "delete from stu where id = ? ";  // 得到连接对象  Connection conn = JdbcUtil.*getConnection*();  // 创建PreparedStatement  PreparedStatement ps  = **null**;  ResultSet rs = **null**;  **int** row = 0;  **try** {  ps = conn.prepareStatement(sql);  // 设置参数并执行  ps.setInt(1, id);  // 执行删除操作  row = ps.executeUpdate();  System.***out***.println("删除成功");  } **catch** (SQLException e) {  e.printStackTrace();  }  **finally** {  JdbcUtil.*close*(conn, ps, rs);  }  **return** row;  } |

## 训练案例4

### 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

### 操作步骤答案

|  |
| --- |
| // 1.编写方法，查询学生表所有的学生，封装成List<Student>返回  **public** **static** **void** main(String[] args)  {  List<Student> students = *getStudents*();  **for** (Student student : students) {  System.***out***.println(student);  }  }    **public** **static** List<Student> getStudents() {  String sql = "select id,name,gender from stu";  //1.得到连接对象  Connection conn = JdbcUtil.*getConnection*();  //2. 创建PreparedStatement  PreparedStatement ps  = **null**;  ResultSet rs = **null**;  List<Student> students = **new** ArrayList<>();  **try** {  ps = conn.prepareStatement(sql);  //3. 因为没有参数，所以直接运行  rs = ps.executeQuery();  **while**(rs.next()) {  Student s = **new** Student();  //封装数据  s.setId(rs.getInt("id"));  s.setName(rs.getString("name"));  s.setGender(rs.getString("gender"));  students.add(s);  }  } **catch** (SQLException e) {  e.printStackTrace();  }  **finally** {  JdbcUtil.*close*(conn, ps, rs);  }  **return** students;  } |

# 关卡3

## 训练案例1

### 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

### 操作步骤答案

|  |
| --- |
| **public** **static** **void** main(String[] args)  **throws** Exception{  // 创建表  // createTable();    // 添加部门数据  // addDept();    // 添加员工数据  // addEmployees();    // 修改id=2的员工工资为16000  // updateSalary(2, 16000);    // 获得所有员工信息  /\* List<Employee> employees = findAllEmployee();  for (Employee employee : employees) {  System.out.println(employee);  } \*/    /\*  Employee e = findEmployeeByName("林青");  System.out.println(e);  \*/    // 删除name为林青霞的员工  // System.out.println(deleteByName("林青霞"));    // 7.编写方法查询所有姓张员工的工资并输出在控制台  // querySalarys();    // 8.查询工资大于10000的员工  /\*List<Employee> employees = findEmployeeBySalary(20000);  for (Employee employee : employees) {  System.out.println(employee);  }\*/    // 9.编写方法查询指定部门的所有员工信息，返回List<Employee>集合  List<Employee> employees = *findEmployeeByDeptName*("产品部");  **for** (Employee employee : employees) {  System.***out***.println(employee);  }  }    // 9.编写方法查询指定部门的所有员工信息，返回List<Employee>集合  **public** **static** List<Employee> findEmployeeByDeptName(String deptName) {  String sql = "select \* from employee where deptid in("  + "select id from dept where dept\_name = ?"  + ");";  //1.得到连接对象  Connection conn = JdbcUtil.*getConnection*();  //2. 创建PreparedStatement  PreparedStatement ps  = **null**;  ResultSet rs = **null**;  List<Employee> employees = **new** ArrayList<>();  **try** {  //2. 创建PreparedStatement  ps = conn.prepareStatement(sql);  //3.设置参数并运行      ps.setString(1, deptName);  rs = ps.executeQuery();  **while**(rs.next()) {  // 创建员工对象  Employee e = **new** Employee(rs.getInt("id"), rs.getString("name"), rs.getString("gender"),  rs.getString("post"), rs.getDouble("salary"), rs.getDate("edate"),  rs.getInt("deptid"));  // 添加到集合中  employees.add(e);  }  } **catch** (SQLException e) {  e.printStackTrace();  }  **finally** {  JdbcUtil.*close*(conn, ps, rs);  }  **return** employees;  }    // 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。  **public** **static** List<Employee> findEmployeeBySalary(**double** salary) {  String sql = "select \* from employee where salary >= ?";  //1.得到连接对象  Connection conn = JdbcUtil.*getConnection*();  //2. 创建PreparedStatement  PreparedStatement ps  = **null**;  ResultSet rs = **null**;  List<Employee> employees = **new** ArrayList<>();  **try** {  //2. 创建PreparedStatement  ps = conn.prepareStatement(sql);  //3.设置参数并运行  ps.setDouble(1, salary);  rs = ps.executeQuery();  **while**(rs.next()) {  // 创建员工对象  Employee e = **new** Employee(rs.getInt("id"), rs.getString("name"), rs.getString("gender"),  rs.getString("post"), rs.getDouble("salary"), rs.getDate("edate"),  rs.getInt("deptid"));  // 添加到集合中  employees.add(e);  }  } **catch** (SQLException e) {  e.printStackTrace();  }  **finally** {  JdbcUtil.*close*(conn, ps, rs);  }  **return** employees;  }    // 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：  **public** **static** **void**  querySalarys() {  String sql = "select salary,name from employee where name like'张%' ";  //1.得到连接对象  Connection conn = JdbcUtil.*getConnection*();  PreparedStatement ps  = **null**;  ResultSet rs = **null**;  **try** {  //2. 创建PreparedStatement  ps = conn.prepareStatement(sql);  //3. 执行查询操作  rs = ps.executeQuery();  **while**(rs.next()) {  System.***out***.println(rs.getString("name") + "=" + rs.getDouble("salary"));  }  } **catch** (SQLException e) {  e.printStackTrace();  } **finally** {  JdbcUtil.*close*(conn, ps,rs);  }  }    // 6.编写方法根据员工姓名删除指定的员工信息。  **public** **static** **boolean** deleteByName(String name){  String sql = "delete from employee where name = ?";  //1.得到连接对象  Connection conn = JdbcUtil.*getConnection*();  PreparedStatement ps  = **null**;  **try** {  //2. 创建PreparedStatement  ps = conn.prepareStatement(sql);  //3.设置参数并运行  ps.setString(1, name);  **int** row = ps.executeUpdate();  **return** row == 1;  } **catch** (SQLException e) {  e.printStackTrace();  } **finally** {  JdbcUtil.*close*(conn, ps);  }  **return** **false**;  }    // 5.编写方法查询指定姓名的员工信息，返回Employee对象。  **public** **static** Employee findEmployeeByName(String name){  String sql = "select \* from employee where name = ?";  //1.得到连接对象  Connection conn = JdbcUtil.*getConnection*();  PreparedStatement ps  = **null**;  ResultSet rs = **null**;  **try** {  //2. 创建PreparedStatement  ps = conn.prepareStatement(sql);  //3.设置参数并运行  ps.setString(1, name);  rs = ps.executeQuery();  rs.next();  // 创建员工对象  Employee e = **new** Employee(rs.getInt("id"), rs.getString("name"), rs.getString("gender"),  rs.getString("post"), rs.getDouble("salary"), rs.getDate("edate"),  rs.getInt("deptid"));  **return** e;  } **catch** (SQLException e) {  e.printStackTrace();  } **finally** {  JdbcUtil.*close*(conn, ps, rs);  }  **return** **null**;  }    // 4.编写方法查询指定职位所有员工的信息，返回List<Employee>集合。  **public** **static** List<Employee> findAllEmployee() {  String sql = "select \* from employee";  //1.得到连接对象  Connection conn = JdbcUtil.*getConnection*();  //2. 创建PreparedStatement  PreparedStatement ps  = **null**;  ResultSet rs = **null**;  List<Employee> employees = **new** ArrayList<>();  **try** {  ps = conn.prepareStatement(sql);  //3. 因为没有参数，所以直接运行  rs = ps.executeQuery();  **while**(rs.next()) {  // 创建员工对象  Employee e = **new** Employee(rs.getInt("id"), rs.getString("name"), rs.getString("gender"),  rs.getString("post"), rs.getDouble("salary"), rs.getDate("edate"),  rs.getInt("deptid"));  // 添加到集合中  employees.add(e);  }  } **catch** (SQLException e) {  e.printStackTrace();  }  **finally** {  JdbcUtil.*close*(conn, ps, rs);  }  **return** employees;  }    // 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。  **public** **static** **void** updateSalary(**int** id,**double** salary)  **throws** Exception {  // 获得连接对象  Connection conn = JdbcUtil.*getConnection*();  // Sql语句  String sql = "update employee set salary = ? where id = ?;";  // 获得PreparedStatement对象  PreparedStatement ps = conn.prepareStatement(sql);  // 设置参数  ps.setDouble(1, salary);  ps.setInt(2, id);  // 执行SQL语句  **int** row = ps.executeUpdate();  System.***out***.println(row);  // 关闭资源  JdbcUtil.*close*(conn, ps);  }    // 添加多条员工数据  **public** **static** **void** addEmployees() **throws** Exception{  // 获得连接对象  Connection conn = JdbcUtil.*getConnection*();  // SQL语句  String sql = "insert into employee(name,gender,post,salary,edate,deptid) "  + "values('张三','男','Java研发工程师',9000,'2016-10-10',1),"  + "('林青霞','女','销售经理',15000,'2015-06-15',2),"  + "('李四','男','产品经理',10000,'2014-12-11',3),"  + "('张三丰','男','Android研发工程师',8000,'2017-05-26',1);";  // 获得PreparedStatement对象  PreparedStatement ps = conn.prepareStatement(sql);  // 执行SQL语句  **int** row = ps.executeUpdate();  System.***out***.println(row);  // 关闭资源  JdbcUtil.*close*(conn, ps);  }      // 添加多个部门数据  **public** **static** **void** addDept() **throws** Exception{  // 获得连接对象  Connection conn = JdbcUtil.*getConnection*();  // SQL语句  String sql = "insert into dept(dept\_name) values('研发部'),('销售部'),('产品部');";  // 获得PreparedStatement对象  PreparedStatement ps = conn.prepareStatement(sql);  // 执行SQL语句  **int** row = ps.executeUpdate();  System.***out***.println(row);  // 关闭资源  JdbcUtil.*close*(conn, ps);  }    /\*  \*  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)  \*/  **public** **static** **void** createTable() **throws** Exception{  // 创建部门表Sql  String sql01 = "create table dept(id int primary key auto\_increment,"  + "dept\_name varchar(20) unique not null);";  // 创建员工表Sql  String sql02 = "create table employee(id int primary key auto\_increment,"  + "name varchar(20) unique not null,"  + "gender varchar(2),"  + "post varchar(20),"  + "salary double,"  + "edate date,"  + "deptid int,"  + "constraint foreign key (deptid) references dept(id))";    // 获得连接对象  Connection conn = JdbcUtil.*getConnection*();  // 获得PreparedStatement对象  Statement ps = conn.createStatement();  // 批处理：可以同时执行多条sql  ps.addBatch(sql01);  ps.addBatch(sql02);  // 创建部门表和员工表  **int**[] row = ps.executeBatch();  System.***out***.println(row.length);    // 关闭资源  JdbcUtil.*close*(conn, ps);  } |