## Day06\_JDBC连接池&JDBCTemplate

### 0.1学习目标

1、理解JDBC相关概念

2、掌握JDBC连接数据库基本步骤

3、掌握JDBC的CRUD操作

4、能够完成JDBC的事务操作

### 0.2 学习指南

* JDBC相关概念（重要）
* JDBC的CRUD操作（重要）
* JDBC管理事务（重要）

### 第一堂课

本节知识点

1、今日内容

2、JDBC\_概念

3、JDBC\_快速入门

4、DBC各个类详解\_DriverManager\_注册驱动

本节目标

掌握JDBC相关概念以及快速入门案例

理解DriverManager

#### JDBC\_概念

##### 1.1.1知识概述

无

##### 1.1.2视频详情



##### 1.1.3总结与补充

无

##### 1.1.4课堂提问与练习

无

##### 1.1.5习题答案

无

##### 1.1.6视频缺陷

无

##### 1.1.7视频扩展

无

#### JDBC\_概念

##### 1.2.1知识概述

概念：Java DataBase Connectivity  Java 数据库连接， Java语言操作数据库  
        \* JDBC本质：其实是官方（sun公司）定义的一套操作所有关系型数据库的规则，即接口。各个数据库厂商去实现这套接口，提供数据库驱动jar包。我们可以使用这套接口（JDBC）编程，真正执行的代码是驱动jar包中的实现类。

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。

##### 1.2.2视频详情



##### 1.2.3总结与补充

有了JDBC，向各种关系数据发送SQL语句就是一件很容易的事。换言之，有了JDBC API，就不必为访问[Sybase数据库](https://baike.baidu.com/item/Sybase%E6%95%B0%E6%8D%AE%E5%BA%93)专门写一个程序，为访问[Oracle数据库](https://baike.baidu.com/item/Oracle%E6%95%B0%E6%8D%AE%E5%BA%93)又专门写一个程序，或为访问Informix数据库又编写另一个程序等等，程序员只需用JDBC API写一个程序就够了，它可向相应数据库发送SQL调用。同时，将Java语言和JDBC结合起来使程序员不必为不同的平台编写不同的应用程序，只须写一遍程序就可以让它在任何平台上运行，这也是Java语言“编写一次，处处运行”的优势。

##### 1.2.4课堂提问与练习

1、什么是JDBC？

##### 1.2.5习题答案

参考知识概述

##### 1.2.6视频缺陷

无

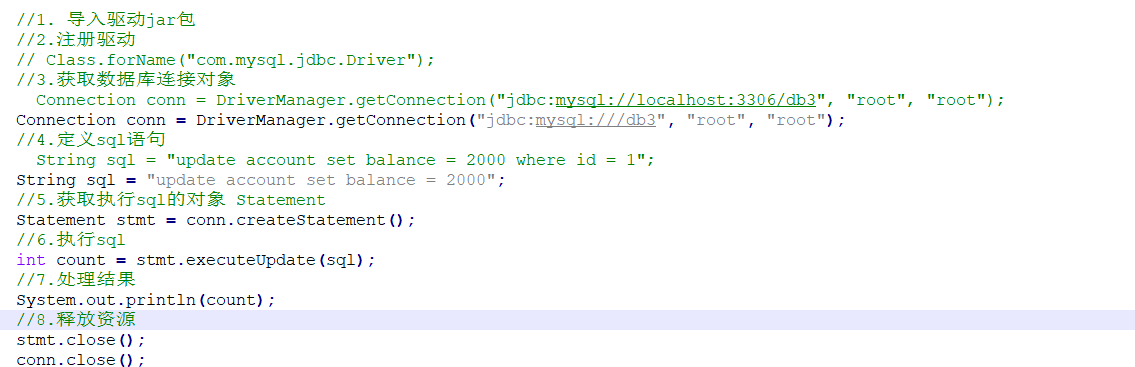
##### 1.2.7视频扩展

无

#### JDBC\_快速入门

##### 1.3.1知识概述

 步骤：  
            1. 导入驱动jar包 mysql-connector-java-5.1.37-bin.jar  
                1.复制mysql-connector-java-5.1.37-bin.jar到项目的libs目录下  
                2.右键-->Add As Library  
            2. 注册驱动  
            3. 获取数据库连接对象 Connection  
            4. 定义sql  
            5. 获取执行sql语句的对象 Statement  
            6. 执行sql，接受返回结果  
            7. 处理结果  
            8. 释放资源



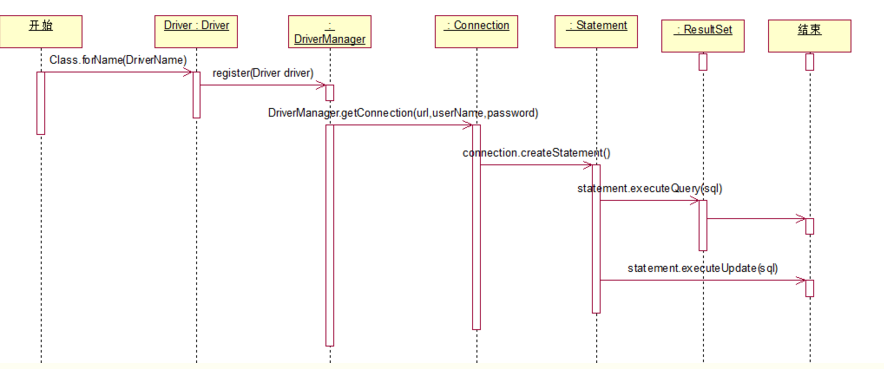
##### 1.3.2视频详情



##### 1.3.3总结与补充

  在操作前必须先获取与数据库的连接。操作完成后必须关闭连接，避免资源浪费。

Jdbc具体操作步骤固定。



##### 1.3.4课堂提问与练习

JDBC的操作步骤

##### 1.3.5习题答案

参考知识概述

##### 1.3.6视频缺陷

无

##### 1.3.7视频扩展

无

#### 1.4JDBC\_快速入门

##### 1.4.1知识概述

功能：  
                1. 注册驱动：告诉程序该使用哪一个数据库驱动jar  
                    static void registerDriver(Driver driver) :注册与给定的驱动程序 DriverManager 。   
                    写代码使用：  Class.forName("com.mysql.jdbc.Driver");  
                    通过查看源码发现：在com.mysql.jdbc.Driver类中存在静态代码块  
                     static {  
                            try {  
                                java.sql.DriverManager.registerDriver(new Driver());  
                            } catch (SQLException E) {  
                                throw new RuntimeException("Can't register driver!");  
                            }  
                        }  
  
                    注意：mysql5之后的驱动jar包可以省略注册驱动的步骤。

##### 1.4.2视频详情



##### 1.4.3总结与补充

  DriverManager 类是 JDBC 的管理层，作用于用户和驱动程序之间。它跟踪可用的驱动程序，并在数据库和相应驱动程序之间建立连接。

DriverManager 类包含一列 Driver 类，它们已通过调用方法 DriverManager.registerDriver 对自己进行了注册。所有 Driver 类都必须包含有一个静态部分。它创建该类的实例，然后在加载该实例时 DriverManager 类进行注册。这样，用户正常情况下将不会直接调用 DriverManager.registerDriver；而是在加载驱动程序时由驱动程序自动调用。

##### 1.4.4课堂提问与练习

无

##### 1.4.5习题答案

无

##### 1.4.6视频缺陷

无

##### 1.4.7视频扩展

无

### 第二堂课

本节知识点：

1JDBC各个类详解\_DriverManager\_获取数据库连接

2 JDBC各个类详解\_Connection

3 JDBC各个类详解\_Statement

4JDBC练习insert语句

本节目标 ：

1、能够理解JDBC各步骤的作用

2、能够完成JDBC的insert操作

#### JDBC各个类详解\_DriverManager\_获取数据库连接

##### 2.1.1知识概述

获取数据库连接：  
                    \* 方法：static Connection getConnection(String url, String user, String password)   
                    \* 参数：  
                        \* url：指定连接的路径  
                            \* 语法：jdbc:mysql://ip地址(域名):端口号/数据库名称  
                            \* 例子：jdbc:mysql://localhost:3306/db3  
                            \* 细节：如果连接的是本机mysql服务器，并且mysql服务默认端口是3306，则url可以简写为：jdbc:mysql:///数据库名称  
                        \* user：用户名  
                        \* password：密码

##### 2.1.2视频详情



##### 2.1.3总结与补充

加载 Driver 类并在 DriverManager 类中注册后，它们即可用来与数据库建立连接。当调用 DriverManager.getConnection 方法发出连接请求时，DriverManager 将检查每个驱动程序，查看它是否可以建立连接。

##### 2.1.4课堂提问与练习

无

##### 2.1.5习题答案

无

##### 2.1.6视频缺陷

无

##### 2.1.7视频扩展

无

#### JDBC各个类详解\_Connection

##### 2.2.1知识概述

Connection：数据库连接对象  
            1. 功能：  
                1. 获取执行sql 的对象  
                    \* Statement createStatement()  
                    \* PreparedStatement prepareStatement(String sql)    
                2. 管理事务：  
                    \* 开启事务：setAutoCommit(boolean autoCommit) ：调用该方法设置参数为false，即开启事务  
                    \* 提交事务：commit()   
                    \* 回滚事务：rollback()

##### 2.2.2视频详情



##### 2.2.3总结与补充

一次Drivermanager.getConnection(jdbcurl)获得只是一个connection

他可以获取sql的执行对象，还可以开启、回滚、提交事务。

Mysql的连接路径为：jdbc:mysql://localhost:3306/db3，user,password

##### 2.2.4课堂提问与练习

无

##### 2.2.5习题答案

无

##### 2.2.6视频缺陷

无

##### 2.2.7视频扩展

无

#### JDBC各个类详解\_Statement

##### 2.3.1知识概述

 执行sql  
                1. boolean execute(String sql) ：可以执行任意的sql 了解   
                2. int executeUpdate(String sql) ：执行DML（insert、update、delete）语句、DDL(create，alter、drop)语句  
                    \* 返回值：影响的行数，可以通过这个影响的行数判断DML语句是否执行成功 返回值>0的则执行成功，反之，则失败。  
                3. ResultSet executeQuery(String sql)  ：执行DQL（select)语句

##### 2.3.2视频详情



##### 2.3.3总结与补充

Statement安全性较低，不能有效防止sql注入问题，项目中很少使用。

Statement 接口提供了三种执行 SQL 语句的方法：executeQuery、executeUpdate 和 execute。使用哪一个方法由 SQL 语句所产生的内容决定。   
  
方法executeQuery   
用于产生单个结果集的语句，例如 SELECT 语句。 被使用最多的执行 SQL 语句的方法是 executeQuery。这个方法被用来执行 SELECT 语句，它几乎是使用最多的 SQL 语句。   
  
方法executeUpdate   
用于执行 INSERT、UPDATE 或 DELETE 语句以及 SQL DDL（数据定义语言）语句，例如 CREATE TABLE 和 DROP TABLE。INSERT、UPDATE 或 DELETE 语句的效果是修改表中零行或多行中的一列或多列。executeUpdate 的返回值是一个整数，指示受影响的行数（即更新计数）。对于 CREATE TABLE 或 DROP TABLE 等不操作行的语句，executeUpdate 的返回值总为零。

##### 2.3.4课堂提问与练习

无

##### 2.3.5习题答案

无

##### 2.3.6视频缺陷

无

##### 2.3.7视频扩展

无

#### 2.4 JDBC练习insert语句

##### 2.4.1知识概述

代码：  
                    Statement stmt = null;  
                    Connection conn = null;  
                    try {  
                        //1. 注册驱动  
                        Class.forName("com.mysql.jdbc.Driver");  
                        //2. 定义sql  
                        String sql = "insert into account values(null,'王五',3000)";  
                        //3.获取Connection对象  
                        conn = DriverManager.getConnection("jdbc:mysql:///db3", "root", "root");  
                        //4.获取执行sql的对象 Statement  
                        stmt = conn.createStatement();  
                        //5.执行sql  
                        int count = stmt.executeUpdate(sql);//影响的行数  
                        //6.处理结果  
                        System.out.println(count);  
                        if(count > 0){  
                            System.out.println("添加成功！");  
                        }else{  
                            System.out.println("添加失败！");  
                        }  
              
                    } catch (ClassNotFoundException e) {  
                        e.printStackTrace();  
                    } catch (SQLException e) {  
                        e.printStackTrace();  
                    }finally {  
                        //stmt.close();  
                        //7. 释放资源  
                        //避免空指针异常  
                        if(stmt != null){  
                            try {  
                                stmt.close();  
                            } catch (SQLException e) {  
                                e.printStackTrace();  
                            }  
                        }  
              
                        if(conn != null){  
                            try {  
                                conn.close();  
                            } catch (SQLException e) {  
                                e.printStackTrace();  
                            }  
                        }  
                    }

##### 2.4.2视频详情



##### 2.4.3总结与补充

需要注意关闭connection,statement,关闭这两个资源的时候要先判断这两个值是否为空。

不为空才可以关闭，为空要抛异常。

##### 2.4.4课堂提问与练习

* 练习JDBC\_insert操作

##### 2.4.5习题答案

参考知识概述

##### 2.4.6视频缺陷

无

##### 2.4.7视频扩展

无

### 第三堂课

本节知识点

1 JDBC练习\_update语句

2 JDBC练习\_update\_DDL语句

3 JDBC各个类详解\_ResultSet\_基本使用

4 JDBC各个类详解\_ResultSet\_遍历结果集

本节目标

1、能够完成JDBC的update操作

2、能够完成JDBC的ReSultSet操作

#### JDBC练习\_update语句

##### 3.1.1知识概述

public static void main(String[] args) {

Connection conn = null;

Statement stmt = null;

try {

//1. 注册驱动

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

//2.获取连接对象

conn = DriverManager.getConnection("jdbc:mysql:///db3", "root", "root");

//3.定义sql

String sql = "update account set balance = 1500 where id = 3";

//4.获取执行sql对象

stmt = conn.createStatement();

//5.执行sql

int count = stmt.executeUpdate(sql);

//6.处理结果

System.out.println(count);

if(count > 0){

System.out.println("修改成功！");

}else{

System.out.println("修改失败");

}

} catch (ClassNotFoundException e) {

e.printStackTrace();

} catch (SQLException e) {

e.printStackTrace();

} finally {

//7.释放资源

if(stmt != null){

try {

stmt.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

if(conn != null){

try {

conn.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

##### 3.1.2视频详情



##### 3.1.3总结与补充

当完成update操作时，要记得关闭连接connection与statement。并且使用的是executeUpdate(sql)方法。

##### 3.1.4课堂提问与练习

练习JDBC\_update语句

##### 3.1.5习题答案

参考知识概述

##### 3.1.6视频缺陷

无

##### 3.1.7视频扩展

无

#### 3.2JDBC练习\_update\_DDL语句

##### 3.2.1知识概述

删除一条记录：

public static void main(String[] args) {

Connection conn = null;

Statement stmt = null;

try {

//1. 注册驱动

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

//2.获取连接对象

conn = DriverManager.getConnection("jdbc:mysql:///db3", "root", "root");

//conn = JDBCUtils.getConnection("jdbc:mysql:///db3", "root", "root");

//3.定义sql

String sql = "delete from account where id = 3";

//4.获取执行sql对象

stmt = conn.createStatement();

//5.执行sql

int count = stmt.executeUpdate(sql);

//6.处理结果

System.out.println(count);

if(count > 0){

System.out.println("删除成功！");

}else{

System.out.println("删除失败");

}

} catch (ClassNotFoundException e) {

e.printStackTrace();

} catch (SQLException e) {

e.printStackTrace();

} finally {

//7.释放资源

if(stmt != null){

try {

stmt.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

if(conn != null){

try {

conn.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

创建一张表：

public static void main(String[] args) {

Connection conn = null;

Statement stmt = null;

try {

//1. 注册驱动

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

//2.获取连接对象

conn = DriverManager.getConnection("jdbc:mysql:///db3", "root", "root");

//3.定义sql

String sql = "create table student (id int , name varchar(20))";

//4.获取执行sql对象

stmt = conn.createStatement();

//5.执行sql

int count = stmt.executeUpdate(sql);

//6.处理结果

System.out.println(count);

} catch (ClassNotFoundException e) {

e.printStackTrace();

} catch (SQLException e) {

e.printStackTrace();

} finally {

//7.释放资源

if(stmt != null){

try {

stmt.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

if(conn != null){

try {

conn.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

##### 3.2.2视频详情



##### 3.2.3总结与补充

注意这里需要使用statement的executeUpdate(sql),方法，并关闭连接connection与statement

##### 3.2.4课堂提问与练习

* 练习JDBC\_update\_DDL操作

##### 3.2.5习题答案

参考知识概述

##### 3.2.6视频缺陷

无

##### 3.2.7视频扩展

无

#### 3.3JDBC各个类详解\_ResultSet\_基本使用

##### 3.3.1知识概述

ResultSet：结果集对象,封装查询结果  
            \* boolean next(): 游标向下移动一行，判断当前行是否是最后一行末尾(是否有数据)，如果是，则返回false，如果不是则返回true  
            \* getXxx(参数):获取数据  
                \* Xxx：代表数据类型   如： int getInt() ,    String getString()  
                \* 参数：  
                    1. int：代表列的编号,从1开始   如： getString(1)  
                    2. String：代表列名称。 如： getDouble("balance")  
              
            \* 注意：  
                \* 使用步骤：  
                    1. 游标向下移动一行  
                    2. 判断是否有数据  
                    3. 获取数据

##### 3.3.2视频详情



##### 3.3.3总结与补充

ResultSet:结果集，封装了使用JDBC进行查询的结果

\* 1.调用Statement对象的excuteQuery(sql)方法可以得到结果集

\* 2.ResultSet返回的实际上就是一张数据表，有一个指针

\* 指向数据表的第一样的前面，可以调用next()方法检测下一行是否有效，若有效则返回true

\* ,并且指针下移，相当于迭代器对象的hasNext()和next()的结合体

\* 3.当指针对位到确定的一行时，可以通过调用getXxx(index)或者getXxx(columnName)

\* 获取每一列的值，例如：getInt(1),getString("name")

\* 4.ResultSet当然也需要进行关闭

##### 3.3.4课堂提问与练习

* 完成ResultSet的基本使用

##### 3.3.5习题答案

public static void main(String[] args) {

Connection conn = null;

Statement stmt = null;

ResultSet rs = null;

try {

//1. 注册驱动

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

//2.获取连接对象

conn = DriverManager.getConnection("jdbc:mysql:///db3", "root", "root");

//3.定义sql

String sql = "select \* from account";

//4.获取执行sql对象

stmt = conn.createStatement();

//5.执行sql

rs = stmt.executeQuery(sql);

//6.处理结果

//6.1 让游标向下移动一行

rs.next();

//6.2 获取数据

int id = rs.getInt(1);

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

double balance = rs.getDouble(3);

System.out.println(id + "---" + name + "---" + balance);

//6.1 让游标向下移动一行

rs.next();

//6.2 获取数据

int id2 = rs.getInt(1);

String name2 = rs.getString("name");

double balance2 = rs.getDouble(3);

System.out.println(id2 + "---" + name2 + "---" + balance2);

//6.1 让游标向下移动一行

rs.next();

//6.2 获取数据

int id3 = rs.getInt(1);

String name3 = rs.getString("name");

double balance3 = rs.getDouble(3);

System.out.println(id3 + "---" + name3 + "---" + balance3);

} catch (ClassNotFoundException e) {

e.printStackTrace();

} catch (SQLException e) {

e.printStackTrace();

} finally {

//7.释放资源

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();

}

}

}

}

##### 3.3.6视频缺陷

无

##### 3.3.7视频扩展

无

#### 3.4JDBC各个类详解\_ResultSet\_遍历结果集

##### 3.4.1知识概述

 使用步骤：  
                    1. 游标向下移动一行  
                    2. 判断是否有数据  
                    3. 获取数据  
  
                   //循环判断游标是否是最后一行末尾。  
                    while(rs.next()){  
                        //获取数据  
                        //6.2 获取数据  
                        int id = rs.getInt(1);  
                        String name = rs.getString("name");  
                        double balance = rs.getDouble(3);  
          
                        System.out.println(id + "---" + name + "---" + balance);  
                    }

##### 3.4.2视频详情



##### 3.4.3总结与补充

遍历时首先要判断是否有数据，遍历完成后要关闭resultSet

##### 3.4.4课堂提问与练习

* 练习JDBC\_ResultSet\_遍历结果集

##### 3.4.5习题答案

public static void main(String[] args) {

Connection conn = null;

Statement stmt = null;

ResultSet rs = null;

try {

//1. 注册驱动

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

//2.获取连接对象

conn = DriverManager.getConnection("jdbc:mysql:///db3", "root", "root");

//3.定义sql

String sql = "select \* from account";

//4.获取执行sql对象

stmt = conn.createStatement();

//5.执行sql

rs = stmt.executeQuery(sql);

//6.处理结果

//循环判断游标是否是最后一行末尾。

while(rs.next()){

//获取数据

//6.2 获取数据

int id = rs.getInt(1);

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

double balance = rs.getDouble(3);

System.out.println(id + "---" + name + "---" + balance);

}

/\* //6.1 让游标向下移动一行

if(rs.next()){

//判断是否有数据

//6.2 获取数据

int id = rs.getInt(1);

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

double balance = rs.getDouble(3);

System.out.println(id + "---" + name + "---" + balance);

}

//6.1 让游标向下移动一行

if(rs.next()){

//判断是否有数据

//6.2 获取数据

int id = rs.getInt(1);

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

double balance = rs.getDouble(3);

System.out.println(id + "---" + name + "---" + balance);

}

//6.1 让游标向下移动一行

if(rs.next()){

//判断是否有数据

//6.2 获取数据

int id = rs.getInt(1);

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

double balance = rs.getDouble(3);

System.out.println(id + "---" + name + "---" + balance);

}

//6.1 让游标向下移动一行

if(rs.next()){

//判断是否有数据

//6.2 获取数据

int id = rs.getInt(1);

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

double balance = rs.getDouble(3);

System.out.println(id + "---" + name + "---" + balance);

}\*/

/\* //6.1 让游标向下移动一行

rs.next();

//6.2 获取数据

int id2 = rs.getInt(1);

String name2 = rs.getString("name");

double balance2 = rs.getDouble(3);

System.out.println(id2 + "---" + name2 + "---" + balance2);

//6.1 让游标向下移动一行

rs.next();

//6.2 获取数据

int id3 = rs.getInt(1);

String name3 = rs.getString("name");

double balance3 = rs.getDouble(3);

System.out.println(id3 + "---" + name3 + "---" + balance3);\*/

} catch (ClassNotFoundException e) {

e.printStackTrace();

} catch (SQLException e) {

e.printStackTrace();

} finally {

//7.释放资源

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();

}

}

}

}

##### 3.4.6视频缺陷

无

##### 3.4.7视频扩展

无

### 第四堂课

本节知识点：

1. JDBC练习\_select语句

本节目标 ：

能够完成JDBC的select语句练习

#### JDBC练习\_select语句

##### 4.1.1知识概述

首先创建一个pojo类Emp，然后使用jdbc的resultSet来接收查询结果。然后把对应列的数据set到EMP的属性名称中。

##### 4.1.2视频详情



##### 4.1.3总结与补充

实体类中的属性名称与类型要与数据库中对应的列相匹配。并且查询完成后要关闭ResultSet、statement、connection等。

##### 4.1.4课堂提问与练习

1.实体类中为什么要重写toString()方法

2.BC练习\_select语句

##### 4.1.5习题答案

1 .Object类的toString()方法总是返回对象的所属类的类名 + @ + hashCode值，代表对象在内存的位置。这显然不能满足我们通常的需求。像这里，我们是希望能打印出EMP的全名出来，这时，就需要重写toString()方法，因为重写了toString()之后，那么EMP在调用toString()方法的时候，会优先调用自己类里的toString()方法。.

2.实体类创建:

private int id;

private String ename;

private int job\_id;

private int mgr;

private Date joindate;

private double salary;

private double bonus;

private int dept\_id;

并提供getset与toString 方法

查询实现：

public static void main(String[] args) {

List<Emp> list = new JDBCDemo8().findAll2();

System.out.println(list);

System.out.println(list.size());

}并

/\*\*

\* 查询所有emp对象

\* @return

\*/

public List<Emp> findAll(){

Connection conn = null;

Statement stmt = null;

ResultSet rs = null;

List<Emp> list = null;

try {

//1.注册驱动

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

//2.获取连接

conn = DriverManager.getConnection("jdbc:mysql:///db3", "root", "root");

//3.定义sql

String sql = "select \* from emp";

//4.获取执行sql的对象

stmt = conn.createStatement();

//5.执行sql

rs = stmt.executeQuery(sql);

//6.遍历结果集，封装对象，装载集合

Emp emp = null;

list = new ArrayList<Emp>();

while(rs.next()){

//获取数据

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

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

int job\_id = rs.getInt("job\_id");

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

Date joindate = rs.getDate("joindate");

double salary = rs.getDouble("salary");

double bonus = rs.getDouble("bonus");

int dept\_id = rs.getInt("dept\_id");

// 创建emp对象,并赋值

emp = new Emp();

emp.setId(id);

emp.setEname(ename);

emp.setJob\_id(job\_id);

emp.setMgr(mgr);

emp.setJoindate(joindate);

emp.setSalary(salary);

emp.setBonus(bonus);

emp.setDept\_id(dept\_id);

//装载集合

list.add(emp);

}

} catch (ClassNotFoundException e) {

e.printStackTrace();

} catch (SQLException e) {

e.printStackTrace();

}finally {

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();

}

}

}

return list;

}

##### 4.1.6视频缺陷

无

##### 4.1.7视频扩展

无

### 第五堂课

本节知识点：

1、JDBC工具类

2、JDBC练习\_登录案例

本节目标 ：

1、能够完成JDBC工具类的编写

2、能够使用JDBC工具类完成登录案例

#### JDBC工具类

##### 5.1.1知识概述

目的：简化书写  
    \* 分析：  
        1. 注册驱动也抽取  
        2. 抽取一个方法获取连接对象  
            \* 需求：不想传递参数（麻烦），还得保证工具类的通用性。  
            \* 解决：配置文件  
                jdbc.properties  
                    url=  
                    user=  
                    password=  
  
  
        3. 抽取一个方法释放资源

##### 5.1.2视频详情



##### 5.1.3总结与补充

每次连接数据库都要重复写jdbc的连接步骤，为了简化开发需要把固定不变的代码抽取出来，这样使用的时候直接调用工具类并把参数传入就可以完成连接数据库的操作了。

##### 5.1.4课堂提问与练习

* 1、编写代码实现JDBC工具类

##### 5.1.5习题答案

public class JDBCUtils {

private static String url;

private static String user;

private static String password;

private static String driver;

/\*\*

\* 文件的读取，只需要读取一次即可拿到这些值。使用静态代码块

\*/

static{

//读取资源文件，获取值。

try {

//1. 创建Properties集合类。

Properties pro = new Properties();

//获取src路径下的文件的方式--->ClassLoader 类加载器

ClassLoader classLoader = JDBCUtils.class.getClassLoader();

URL res = classLoader.getResource("jdbc.properties");

String path = res.getPath();

// System.out.println(path);///D:/IdeaProjects/itcast/out/production/day04\_jdbc/jdbc.properties

//2. 加载文件

// pro.load(new FileReader("D:\\IdeaProjects\\itcast\\day04\_jdbc\\src\\jdbc.properties"));

pro.load(new FileReader(path));

//3. 获取数据，赋值

url = pro.getProperty("url");

user = pro.getProperty("user");

password = pro.getProperty("password");

driver = pro.getProperty("driver");

//4. 注册驱动

Class.forName(driver);

} catch (IOException e) {

e.printStackTrace();

} catch (ClassNotFoundException e) {

e.printStackTrace();

}

}

/\*\*

\* 获取连接

\* @return 连接对象

\*/

public static Connection getConnection() throws SQLException {

return DriverManager.getConnection(url, user, password);

}

/\*\*

\* 释放资源

\* @param stmt

\* @param conn

\*/

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

if( stmt != null){

try {

stmt.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

if( conn != null){

try {

conn.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

/\*\*

\* 释放资源

\* @param stmt

\* @param conn

\*/

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

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();

}

}

}

##### 5.1.6视频缺陷

无

##### 5.1.7视频扩展

无

#### JDBC练习\_登录案例

##### 5.2.1知识概述

 需求：  
            1. 通过键盘录入用户名和密码  
            2. 判断用户是否登录成功  
                \* select \* from user where username = "" and password = "";  
                \* 如果这个sql有查询结果，则成功，反之，则失败  
  
        \* 步骤：  
            1. 创建数据库表 user  
                CREATE TABLE USER(  
                    id INT PRIMARY KEY AUTO\_INCREMENT,  
                    username VARCHAR(32),  
                    PASSWORD VARCHAR(32)  
                  
                );  
  
                INSERT INTO USER VALUES(NULL,'zhangsan','123');  
                INSERT INTO USER VALUES(NULL,'lisi','234');  
  
            2. 代码实现：  
                public class JDBCDemo9 {  
  
                    public static void main(String[] args) {  
                        //1.键盘录入，接受用户名和密码  
                        Scanner sc = new Scanner(System.in);  
                        System.out.println("请输入用户名：");  
                        String username = sc.nextLine();  
                        System.out.println("请输入密码：");  
                        String password = sc.nextLine();  
                        //2.调用方法  
                        boolean flag = new JDBCDemo9().login(username, password);  
                        //3.判断结果，输出不同语句  
                        if(flag){  
                            //登录成功  
                            System.out.println("登录成功！");  
                        }else{  
                            System.out.println("用户名或密码错误！");  
                        }  
                    }  
  
                    /\*\*  
                     \* 登录方法  
                     \*/  
                    public boolean login(String username ,String password){  
                        if(username == null || password == null){  
                            return false;  
                        }  
                        //连接数据库判断是否登录成功  
                        Connection conn = null;  
                        Statement stmt =  null;  
                        ResultSet rs = null;  
                        //1.获取连接  
                        try {  
                            conn =  JDBCUtils.getConnection();  
                            //2.定义sql  
                            String sql = "select \* from user where username = '"+username+"' and password = '"+password+"' ";  
                            //3.获取执行sql的对象  
                            stmt = conn.createStatement();  
                            //4.执行查询  
                            rs = stmt.executeQuery(sql);  
                            //5.判断  
                           /\* if(rs.next()){//如果有下一行，则返回true  
                                return true;  
                            }else{  
                                return false;  
                            }\*/  
                           return rs.next();//如果有下一行，则返回true  
                  
                        } catch (SQLException e) {  
                            e.printStackTrace();  
                        }finally {  
                            JDBCUtils.close(rs,stmt,conn);  
                        }  
                        return false;  
                    }  
                }

##### 5.2.2视频详情



##### 5.2.3总结与补充

用到了scanner类，需要进行2次数据，一次账号一次密码。拿到输入的数据放入到sql语句中作为条件进行查询，如果数据不为空，则登录成功。

##### 5.2.4课堂提问与练习

* 1、编写代码实现JDBC\_登录案例

##### 5.2.5习题答案

参考知识概述。

##### 5.2.6视频缺陷

无

##### 5.2.7视频扩展

无

### 第六堂课

本节知识点：

1、 JDBC各个类详解\_PreparedStatement

2、 JDBC管理事务\_概述

3、 JDBC管理事务\_实现

本节目标 ：

1. 掌握PreparedStatement的使用
2. 完成JDBC事务的实现

#### JDBC各个类详解\_PreparedStatement

##### 6.1.1知识概述

执行sql的对象  
            1. SQL注入问题：在拼接sql时，有一些sql的特殊关键字参与字符串的拼接。会造成安全性问题  
                1. 输入用户随便，输入密码：a' or 'a' = 'a  
                2. sql：select \* from user where username = 'fhdsjkf' and password = 'a' or 'a' = 'a'   
  
            2. 解决sql注入问题：使用PreparedStatement对象来解决  
            3. 预编译的SQL：参数使用?作为占位符  
            4. 步骤：  
                1. 导入驱动jar包 mysql-connector-java-5.1.37-bin.jar  
                2. 注册驱动  
                3. 获取数据库连接对象 Connection  
                4. 定义sql  
                    \* 注意：sql的参数使用？作为占位符。 如：select \* from user where username = ? and password = ?;  
                5. 获取执行sql语句的对象 PreparedStatement  Connection.prepareStatement(String sql)   
                6. 给？赋值：  
                    \* 方法： setXxx(参数1,参数2)  
                        \* 参数1：？的位置编号 从1 开始  
                        \* 参数2：？的值  
                7. 执行sql，接受返回结果，不需要传递sql语句  
                8. 处理结果  
                9. 释放资源

5. 注意：后期都会使用PreparedStatement来完成增删改查的所有操作  
                1. 可以防止SQL注入  
                2. 效率更高

##### 6.1.2视频详情



##### 6.1.3总结与补充

一般使用preparedStatement执行sql语句，因为效率高，还可以有效防止sql注入问题。

在JDBC应用中，如果你已经是稍有水平开发者，你就应该始终以PreparedStatement代替Statement.也就是说，在任何时候都不要使用Statement。

PreparedStatement接口继承[Statement](https://baike.baidu.com/item/Statement)，并与之在两方面有所不同：

PreparedStatement 实例包含已编译的 SQL 语句。这就是使语句“准备好”。包含于 PreparedStatement 对象中的 SQL 语句可具有一个或多个 IN 参数。IN参数的值在 SQL 语句创建时未被指定。相反的，该语句为每个 IN 参数保留一个问号（“？”）作为[占位符](https://baike.baidu.com/item/%E5%8D%A0%E4%BD%8D%E7%AC%A6)。每个问号的值必须在该语句执行之前，通过适当的setXXX 方法来提供。

由于 PreparedStatement 对象已[预编译](https://baike.baidu.com/item/%E9%A2%84%E7%BC%96%E8%AF%91)过，所以其执行速度要快于 Statement 对象。因此，多次执行的 SQL 语句经常创建为 PreparedStatement 对象，以提高效率。

作为 Statement 的子类，PreparedStatement 继承了 Statement 的所有功能。另外它还添加了一整套方法，用于设置发送给数据库以取代 IN 参数[占位符](https://baike.baidu.com/item/%E5%8D%A0%E4%BD%8D%E7%AC%A6)的值。同时，三种方法 execute、 executeQuery 和 executeUpdate 已被更改以使之不再需要参数。这些方法的 Statement 形式（接受 SQL 语句参数的形式）不应该用于 PreparedStatement 对象。

##### 6.1.4课堂提问与练习

编写代码实现PreparedStatement案例

##### 6.1.5习题答案

/\*\*

\* 登录方法,使用PreparedStatement实现

\*/

public boolean login2(String username ,String password){

if(username == null || password == null){

return false;

}

//连接数据库判断是否登录成功

Connection conn = null;

PreparedStatement pstmt = null;

ResultSet rs = null;

//1.获取连接

try {

conn = JDBCUtils.getConnection();

//2.定义sql

String sql = "select \* from user where username = ? and password = ?";

//3.获取执行sql的对象

pstmt = conn.prepareStatement(sql);

//给?赋值

pstmt.setString(1,username);

pstmt.setString(2,password);

//4.执行查询,不需要传递sql

rs = pstmt.executeQuery();

//5.判断

/\* if(rs.next()){//如果有下一行，则返回true

return true;

}else{

return false;

}\*/

return rs.next();//如果有下一行，则返回true

} catch (SQLException e) {

e.printStackTrace();

}finally {

JDBCUtils.close(rs,pstmt,conn);

}

return false;

}

##### 6.1.6视频缺陷

无

##### 6.1.7视频扩展

无

#### JDBC管理事务\_概述

##### 6.2.1知识概述

事务：一个包含多个步骤的业务操作。如果这个业务操作被事务管理，则这多个步骤要么同时成功，要么同时失败。

##### 6.2.2视频详情



##### 6.2.3总结与补充

通过connection开启、关闭、回滚事务。一般增删改需要事务，也就是说只要有需要改动就需要开启事务。

##### 6.2.4课堂提问与练习

无

##### 6.2.5习题答案

无

##### 6.2.6视频缺陷

无

##### 6.2.7视频扩展

无

#### JDBC管理事务\_实现

##### 6.3.1知识概述

操作：  
        1. 开启事务  
        2. 提交事务  
        3. 回滚事务  
  使用Connection对象来管理事务  
        \* 开启事务：setAutoCommit(boolean autoCommit) ：调用该方法设置参数为false，即开启事务  
            \* 在执行sql之前开启事务  
        \* 提交事务：commit()   
            \* 当所有sql都执行完提交事务  
        \* 回滚事务：rollback()   
            \* 在catch中回滚事务

##### 6.3.2视频详情



##### 6.3.3总结与补充

在执行SQL语句之前，先执行start transaction，这就开启了一个事务（事务的起点），然后可以去执行多条SQL语句，最后要结束事务，commit表示提交，即事务中的多条SQL语句所作出的影响会持久到数据库中，或者rollback，表示回滚到事务的起点，之前做的所有操作都被撤销了。

##### 6.3.4课堂提问与练习

* 编写代码实现JDBC管理事务

##### 6.3.5习题答案

public static void main(String[] args) {

Connection conn = null;

PreparedStatement pstmt1 = null;

PreparedStatement pstmt2 = null;

try {

//1.获取连接

conn = JDBCUtils.getConnection();

//开启事务

conn.setAutoCommit(false);

//2.定义sql

//2.1 张三 - 500

String sql1 = "update account set balance = balance - ? where id = ?";

//2.2 李四 + 500

String sql2 = "update account set balance = balance + ? where id = ?";

//3.获取执行sql对象

pstmt1 = conn.prepareStatement(sql1);

pstmt2 = conn.prepareStatement(sql2);

//4. 设置参数

pstmt1.setDouble(1,500);

pstmt1.setInt(2,1);

pstmt2.setDouble(1,500);

pstmt2.setInt(2,2);

//5.执行sql

pstmt1.executeUpdate();

// 手动制造异常

int i = 3/0;

pstmt2.executeUpdate();

//提交事务

conn.commit();

} catch (Exception e) {

//事务回滚

try {

if(conn != null) {

conn.rollback();

}

} catch (SQLException e1) {

e1.printStackTrace();

}

e.printStackTrace();

}finally {

JDBCUtils.close(pstmt1,conn);

JDBCUtils.close(pstmt2,null);

}

}

##### 6.3.6视频缺陷

无

##### 6.3.7视频扩展

无